
öS 60.-/sfr 7.-/Lit 7000/hfl 8,50 **DM7,–**

AMIGA

Markt&Technik

1–'88

DAS COMPUTERMAGAZIN FÜR AMIGA-FANS

Tests und Kaufberatung

Die besten Spiele

Welche ist die richtige?

Programmiersprachen im Vergleich

Unsichtbare Bedrohung

Viren im Amiga

ARD-Fernsehserie

COMPUTERZEIT

Ergänzende Informationen zur Folge 26: »Spiele«

C

MODULA

BASIC

PR

...as AMIGA-Magazin testet:

...d Perfect ★ Page Flipper ★ Landscape ★ Profimat ★ Logic Works ★ NEC P2200



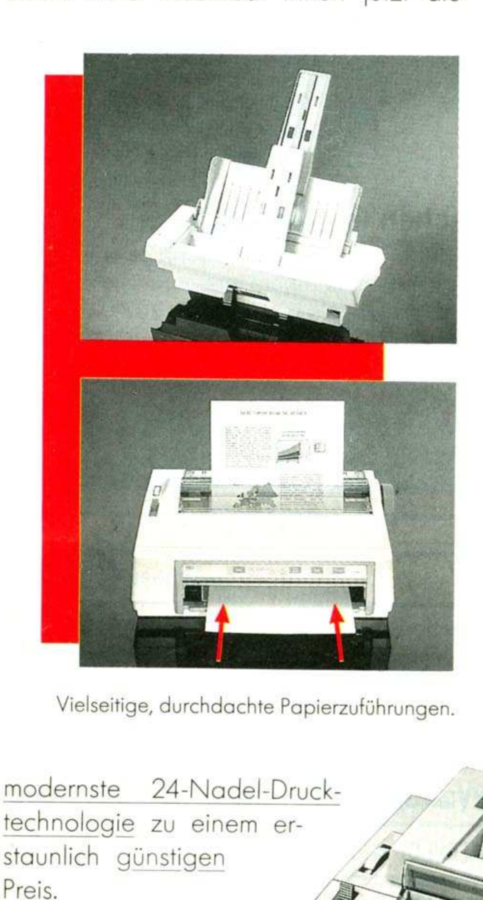
P 2200 – DAS PREIS-LEISTUNGS-GENIE

PROFIQUALITÄT ZUM AMATEURPREIS

EIN NEC DRUCKER FÜR JEDERMANN

Die Computer-Anwender haben Grund zum Jubeln!

Genial – endlich ein Drucker, der für Einsteiger, Aufsteiger und Semiprofis geeignet und vor allem erschwinglich ist. Denn NEC erschließt Ihnen jetzt die modernste 24-Nadel-Drucktechnologie zu einem erstaunlich günstigen Preis.

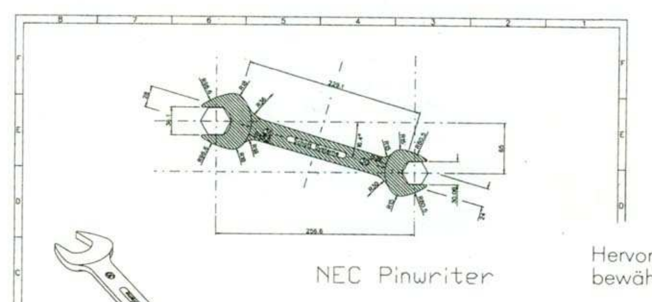


Vielseitige, durchdachte Papierzuführungen.

NEC ist mit seinen 24-Nadel-Druckern in Deutschland marktführend.

Was den P 2200 als echten Profi auszeichnet, sind seine hohe Auflösung von 360 x 360 dpi, ein halbes Dutzend serienmäßiger Schriftarten und eine Reihe praktischer Papierzuführungen. Zum Beispiel können Sie zwischendurch einen Brief drucken, ohne daß das Endlospapier extra herausgenommen werden muß.

Durch seine volle Kompatibilität mit den NEC Pinwritern der 24-Nadel-Serie harmoniert der P 2200 mit allen wichtigen Software-Paketen.



- 1 -

DAS ELEKTRISCHE FELD

Die elektrostatische Kraft zwischen zwei Körpern mit den Ladungen Q_1 und Q_2 kann als Fernwirkung vorgestellt werden, die irgendwie den Abstand zwischen den Körpern überbrückt. Fruchtbarer ist aber die Feldvorstellung: Die Ladung

Hervorragende Druckqualität durch bewährte 24-Nadel-Technologie.

NEC Pinwriter P 2200 mit automatischer Einzelblattführung

Erleben Sie eine neue Dimension: gestochen scharfen Korrespondenzdruck mit ungewöhnlich reicher Schriftartenauswahl, brillante Grafik-Darstellung, bequeme Druckersteuerung und integrierte Papierzuführungen.

Warum also tief in die Taschen greifen, wenn es schon für wenig Geld 24-Nadel-Technologie mit allen Raffinessen gibt?

Endlich braucht niemand mehr auf die bewährte NEC Produkt- und Druckqualität zu verzichten.

Eine breitgefächerte Gruppe – vom Schüler über den Heimanwender bis hin zum Freiberufler – findet im P 2200 die ideale Drucklösung. Anwendern, die schon seit langem auf der Suche nach einem preisgünstigen Drucker für ihren Computer sind, eröffnet er die Möglichkeit, Druckergebnisse in bewährter NEC-Qualität zu erzielen.



12 Schriftartenkassetten zusätzlich erhältlich.

Damit ist der P 2200 die optimale wirtschaftliche und technische Alternative für alle, die sich bei gleichem finanziellen Aufwand bisher nur mit antiquierten 9-Nadel-Druckern begnügen mußten.

Weitere Informationen zum P 2200 erhalten Sie von Ihrem NEC Drucker-Fachhändler.

NEC

NEC Deutschland GmbH

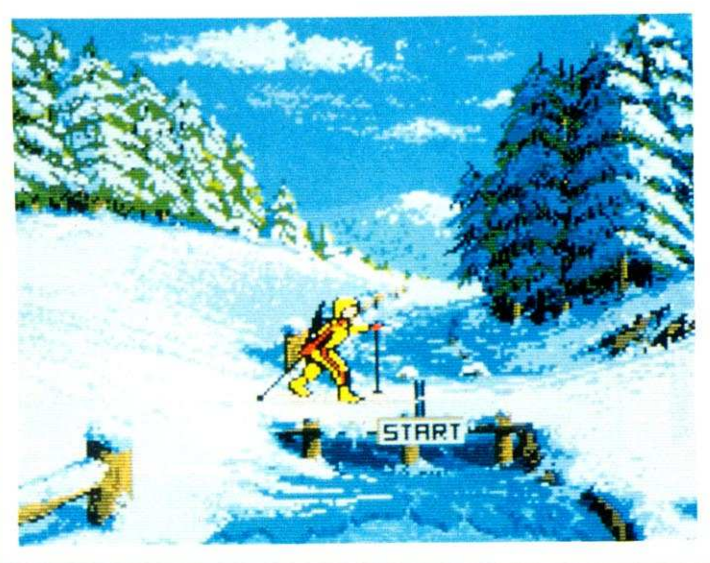
Klausenburger Straße 4, 8000 München 80
Tel.: 0 89/9 30 06-0, Telefax: 0 89/93 77 76/8
Telex: 5 218 073 und 5 218 074 nec m



Action in der Spielecke

Rechtzeitig zu Weihnachten gibt es heiße Informationen für die Spieler unter den Amiga-Fans. Im erweiterten Teil mit Spieletests treten wie immer einige interessante Neuerscheinungen gegeneinander an. Während eine große Übersicht Hilfestellung gibt, um aus der Spieleflut das beste herauszufinden. Einen Bildschirmausschnitt von Wintergames sehen Sie unten.
Ab Seite 26





Programmiersprachen im Vergleich

Auf dem Amiga hat man die Qual der Wahl, wenn es um Programmiersprachen geht. Jede hat ihre eigenen Vorzüge und Nachteile. Wie sie im Vergleich zu sehen sind und wie man sie vereinigen kann, zeigen Ihnen die ausführlichen Tests zu den Sprachen Basic, C, Modula 2, Assembler, Prolog und Lisp. Sehen Sie selbst, welche Sprache aus dem babylonischen Sprachgewirr die richtige für Sie ist.
Ab Seite 18

AKTUELLES

EINSTEIGER

PROGRAMMIERSPRACHEN

SPIELE

KURSE

TIPS & TRICKS

SOFTWARE

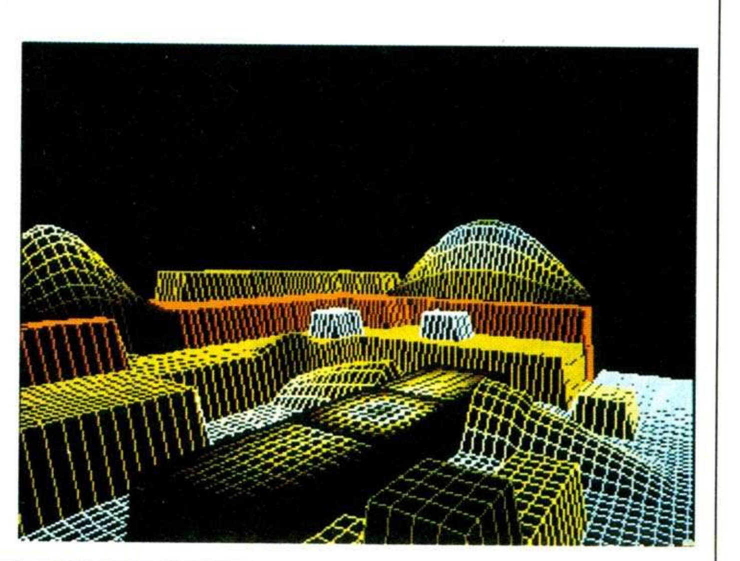
SOFTWARE-TEST

INHALT 1/88

Landschaften aus einer Hand

Wollten Sie sich schon immer einmal eigene Landschaften erschaffen? Berge und Täler von netzgrafikähnlicher Beschaffenheit können Sie jetzt auf dem Amiga mit dem »Landscape Designer« selbst erstellen. Das Programm ist einfach zu bedienen und eignet sich deswegen hervorragend für Einsteiger auf dem Grafiksektor. Lassen Sie Ihrer Fantasie freien Lauf und entwerfen Sie farbige Bit-Gebirge.

Seite 135

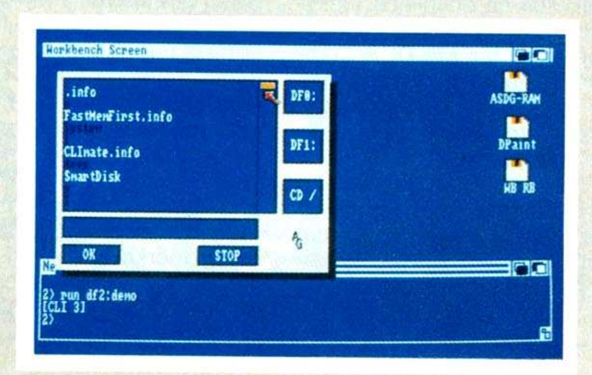


AUFRUFE UND WETTBEWERBE

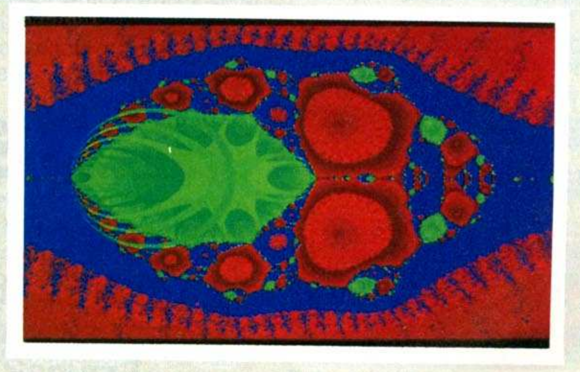
RUBRIKEN

— Der Testbericht P 2200 wurde in die nächste Ausgabe verschoben

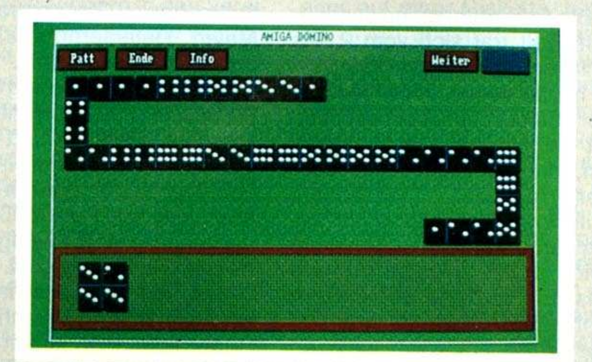
LISTINGS ZUM ABTIPPEN



Einfache Filerequester in C erhalten Sie problemlos mit dem Programm »Disk.h«

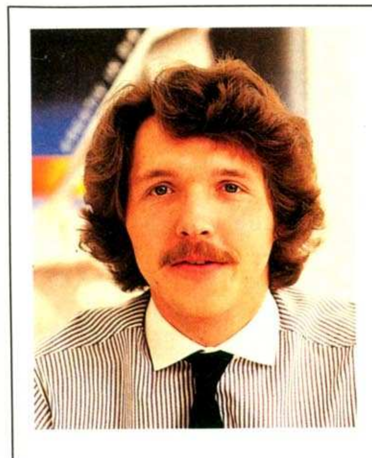


Faszinierende Bilder aus der Juliamenge generiert das Basic-Programm »Julia«



Unser Programm des Monats: »Domino«

Dieses Symbol zeigt an, welche Programme auf Diskette erhältlich sind.



Rollt die Welle?

WordPerfect hat einen wohlklingenden Namen in der PC-Welt, soll es doch das meistverkaufte Textverarbeitungsprogramm für MS-DOS-Computer weltweit sein. Und das bei einer wahren Flut von Konkurrenzprodukten für IBM-PCs und Kompatible.

Dieses mit allen Raffinessen ausgestattete Bearbeitungssystem für Texte verrichtet nun auch auf dem Amiga seinen Dienst. Mit der Reputation und dem guten Klang, welches dieses Programm hat, strahlt der Glanz des Professionellen über dem Amiga-Lattice, Microsoft und andere mögen an dieser Stelle Verständnis zeigen. WordPerfect wurde speziell für den Amiga umgeschrieben. Dennoch blieb die Bedienung des Programms gleich, also hauptsächlich über Alt-, Ctrl- oder Shift-Tastenkombinationen. Dies mag für den einen oder anderen Amiga-Besitzer als Anachronismus erscheinen. Doch es gibt wie immer Vor- und Nachteile. Der Vorteil: die für Anfang des Jahres '88 angekündigten PlanPerfect und DataPerfect werden ähnlich zu steuern sein. Die Bedienung auf anderen Computern funktioniert im Prinzip genauso. Das heißt großartiges Umbeziehungsweise Neulernen ist nicht notwendig. Auf der anderen Seite, da haben Sie einen fantastischen Computer mit moderner Benutzerführung und einer Maus, die sogar im Lieferumfang enthalten ist, und dann kommen Programme mit überragenden Leistungsmerkmalen auf den Markt, die ignorieren diese Bedieneroberfläche einfach. Wie stehen Sie dazu? Wie wichtig ist für Sie der Bedienungskomfort im Verhältnis zur Funktionsvielfalt? Sind Sie der Meinung, das eine schließt das andere aus, oder sollte auf die Bedienung eines Programms aus Sicht einer einheitlichen Produktlinie oder der des Computers geachtet werden?

Ungeachtet dieser Diskussion könnte das Erscheinen von WordPerfect auf dem Amiga eine Art Initialzündung auslösen. Es wird gemunkelt, daß andere Riesen der Softwarebranche wie Borland und Lotus ebenfalls Programme entweder bereits vor der Fertigstellung oder in der Planung haben. Sie verharren derzeit lediglich noch in einer »wait-and-see«-Position. Sie wollen erst abwarten, wie sich der Amiga im Markt durchsetzen kann.

Vorsichtig sollten wir bei einem anderen Thema werden. Viren, erst seit ein paar Jahren ein Gesprächsthema in der Computerszene, machen in letzter Zeit immer mehr von sich reden und mittlerweile ihrem Ziel alle Ehre. Sie verbreiten sich rasend schnell. Um einen Gag oder Spaß handelt es sich, wenn man einem »gutmütigen« Virus begegnet. Da wird der Bildschirm angeknabbert, oder Sie bekommen eine witzige Meldung und ähnliches mehr. Die aggressiven Viren stellen allerdings eine echte Bedrohung der eigenen Datenbestände und der Originalsoftware dar. Sie vernichten oder machen Daten unbrauchbar. Und bei durch Viren zerstörten Originalprogrammen kommt es auf die Kulanz des Händlers oder Herstellers an, ob Ersatz geleistet wird. Abhilfe schaffen hier Wachsamkeit, Vorsicht und Mißtrauen allen unbekannten Programmen gegenüber. Was zu machen ist, erfahren Sie in dieser Ausgabe. Die Prosmiskuität beim Softwaretausch kann auf jeden Fall darunter leiden.

Herzlich Ihr
Albert Absmeier
Chefredakteur



Die Computerszene der ganzen Welt traf sich vom 2. bis 6. November auf der COMDEX 87 in Las Vegas. In der **Ausgabe 1/88** der 68000er berichten wir aktuell über alle Neuigkeiten, die die Hard- und Software-Hersteller vorstellten. Pull-Down-Menüs und ein wenig Farbe alleine machen noch keine gute Benutzerführung aus. Ein Experte auf diesem Gebiet, Professor Frederic Vester, erklärt, nach welchen Gesichtspunkten eine Benutzeroberfläche aufgebaut sein sollte. Außerdem führt Sie ein Kurs in die Grundlagen der Benutzerführung ein. Die Datenfernübertragung gewinnt immer mehr an Popularität bei den Computeranwendern. In der letzten Zeit wächst die Zahl der Mailboxen. Doch nicht jede Box hält, was sie verspricht. Unsere Mailbox-Liste informiert Sie über alle interessanten Mailboxen.

Was nützt dem Computer-Anwender der beste und schnellste Computer ohne gute Software? Ohne Programm ist der Computer eine dumme Kiste und erst die Software haucht ihm das Leben ein. Wir haben ein Resümee aus dem Jahr '87 gezogen und die besten Programme aus den Bereichen Textverarbeitung, Datenbanken, Programmieren, Malen, Datenfernübertragung und Musik ausgewählt.

Der Raytracing-König Sculpt 3D hat einen Konkurrenten bekommen: Silver. Schafft es Silver, den amtierenden Meister vom Thron zu stoßen? In einem ausführlichen Test mußte Silver zeigen, welche Fähigkeiten es dem Amiga entlockt.

Erst mit den Spezialchips wird der Amiga zum schnellen Superrechner. In einem mehrteiligen Kurs lernen Sie, wie man die Spezialchips programmiert und so noch mehr aus dem Amiga herausholt.



Der Amiga ist ein Video- und Grafikprofi. In der **Ausgabe 1/88** testen wir die Amiga-Programme Videoscape 3D, Deluxe Video und Sculpt 3D. Wie leistungsfähig sind sie wirklich? Wer braucht welches Programm? Der Archimedes von Acorn ist ein neuer Computer mit zukunftsweisender Technik. Er ist nicht nur superschnell, sondern bietet auch 8-Kanal-Stereosound und eine maximale Auflösung von 1280 x 1024 Bildpunkten. Ist er eine Konkurrenz für den Amiga? Ein etablierter Konkurrent für den Amiga ist der Atari ST. Happy-Computer will es jetzt wissen: Wer ist besser? Beteiligen Sie sich an diesem Superwettbewerb und zeigen Sie, was der Amiga wirklich kann. Attraktive Geldpreise winken.

Die **Ausgabe 2/88** der Happy-Computer zeigt, wie man anderen Computer-Besitzern digitale Streiche spielt. Tolle Gag-Programme sorgen für viel Spaß. Jetzt ist die richtige Zeit, den Urlaub für den nächsten Sommer zu planen. Eine umfangreiche Übersicht zeigt, welche Computercamps es wo mit welchen Angeboten und Preisen gibt. Wenn Sie sich noch zu Hause entspannen wollen, ist das Amiga-Listing »Flottenmanöver« genau das richtige für Sie. Wenn Sie den Amiga selbst programmieren wollen, hilft Ihnen unser Basic-Kurs. Einfache Beispiele zeigen, wie man durch wenige Programmzeilen viel Wirkung erhält. 24-Nadel-Drucker werden immer preiswerter. Sinkt die Qualität mit dem Preis? Ein Vergleichstest zwischen drei 24-Nadel-Druckern bringt Klarheit. Welche Zukunft haben Btx und DFÜ? Unser Postminister Schwarz-Schilling steht Rede und Antwort. Außerdem stellen wir Btx-Lösungen sowie Btx-fähige Fernseher vor.



Illustration: Rolf Boyke

Virus: Alarmstufe ROT

»Die ich rief, die Geister, werd sie nicht mehr los«. Die ersten Viren auf dem Amiga sorgen für Aufregung. Woher kommen sie? Gibt es einen Schutz?

Großalarm: Ein Virus ist aus einem geheimen Softwarelabor entkommen. In kürzester Zeit hat er weltweit Programme und Computer infiziert. Viele Rechenanlagen sind bereits lahmgelegt. Es gibt kein Gegenmittel.

Science-fiction? Keineswegs: Computerviren bedrohen die Computerwelt. Es sind Programme, die in der Lage sind, sich zu reproduzieren und von Computer zu Computer zu verbreiten. Sie sind noch jung. 1983 berichtete Fred Cohen von der University of Southern California zum ersten Male über derartige Mechanismen.

Der Name Computer-Virus ist angelehnt an einen biologischen Begriff: Viren dringen in Zellkerne ein, manipulieren die Erbsubstanz (DNA), so daß neue Viren produziert werden.

Computer-Viren verhalten sich ähnlich. Sie dringen in Computerprogamme ein und verändern diese. Das infizierte Programm produziert nun ebenfalls Virus-Programme. Als Medium, um sich von Computer zu Computer fortzupflanzen, dienen Disketten oder die direkte Verbindung zweier Rechenanlagen.

Ein Unterschied: Computer-Viren sind nicht durch Evolution entstanden, keine Laune der Natur. Programmierer sind verantwortlich für ihre Entstehung, getrieben von Neugier, Sucht nach zweifelhaftem Ruhm oder Boshaftigkeit.

Die ersten Viren tauchten auf Großrechnern in den Vereinigten Staaten auf. Sie wurden gezielt installiert. Die Gründe sind vielschichtig:

- Sabotage, um einen Konkurrenten auszuschalten
- Rache für eine Entlassung
- oder der Versuch, anderen Computerbesitzern einen Streich zu spielen.

Mit der Heimcomputer-Welle haben die Viren auch Einzug in den privaten Bereich gehalten. Manch einer versucht seinen eigenen Virus zu entwerfen. »Frankenstein spielen auf dem Computer«. Die meisten der Programmierer sind sich der Gefahr für Programme und Daten anderer Computerbesitzer nicht bewußt, die durch die Verbreitung des Virus entsteht. Ein Virus kann verheerende Folgen haben.

Ein Virus-Programm ist kurz. Es schlummert im Speicher oder auf einer Diskette, haftet an einem anderen Programm und wartet. Es wartet auf eine Chance, sich zu vermehren. Sobald das System die »Träger-Routine« aufruft, wird auch der Virus aktiv. Jetzt sucht er sich ein weiteres Programm — auf Diskette, auf der Hard-Disk, im Speicher oder gar auf einem angeschlossenen weiteren Computer — schreibt sich davor, dahinter oder mittenrein.

Das befallene Programm wird länger. Doch Viren wollen unerkannt bleiben, verbergen und schützen sich. Eventuell fälschen sie sogar Einträge im

Inhaltsverzeichnis einer Diskette, um ihre Anwesenheit zu verschleiern.

Auch wenn sich Viren nur kopieren, geht Speicherplatz verloren. Programme werden größer und größer. An jede Routine kann sich ein Virus hängen. Bald belegen nur die Viren Speicherplatz und Rechenzeit. Der befallene Computer kollabiert, ist klinisch tot.

Viele Viren haben noch einen zweiten Auftrag, wollen sich nicht nur verbreiten. Sie machen auf sich aufmerksam, verhöhnen den Benutzer — oder zerstören Dateien.

Manche gezielt installierte Viren machen sich erst nach Jahren bemerkbar. Um den Urheber zu verschleiern, werden sie nicht sofort aktiv. Der Vergleich mit der Inkubationszeit drängt sich auf; eine Zeitbombe im Computer.

In vielen Fällen steckt keine feindliche Intention hinter den Viren. Beispiele gibt es auf allen Computern:

- Spinnen krabbeln über den Bildschirm.
- Kleine gefräßige Tierchen »zernagen« Screens und Windows.

Leben und Löschen

- Manche Viren verlangen vom Bediener eine sofortige Reaktion. Bei einem dieser Exemplare erscheint von Zeit zu Zeit ein kleiner Pac-Man und »schreit« nach Kuchen. Drücken Sie nicht <CTRL C>, »frißt« das liebe Tierchen ein wenig von Ihrem Speicher.

Der Amiga blieb von der Entwicklung nicht verschont. Ein Jahr nach der Auslieferung des Amiga 1000 in der Bundesrepublik wurde ein Virus fertiggestellt und in Umlauf gebracht. Geburtsstätte war die Softwareschmiede des Schweizer Computerclubs »SCA«. Wie uns ein Mitglied dieses Clubs mitteilte, bestand die eigentliche Aufgabe des Virus darin, einige ungeliebte Freunde in Angst und Schrecken zu versetzen und deren Softwaresammlung zu verunstalten.

Der Virus verbreitete sich schneller als geplant. Über Diskettentausch wurde er weitertransportiert. Nach etwa drei Wochen hatte er schon große Teile des Bundesgebietes erfaßt. Nach vier Wochen tauchten erste Exemplare in den USA auf.

Der SCA-Virus ist aufgrund seiner Verbreitung der interessanteste. Er versteckt sich auf den ersten beiden Sektoren im »Bootblock« einer Diskette.

Im Bootblock einer Startdiskette, zum Beispiel Ihrer Workbench-Diskette, befindet sich eine Kennung (DOS) sowie eine DOS-Initialisierungsroutine. Beim Starten führt der Amiga folgende Schritte aus:

- Er erkennt die Startdiskette.
- Er lädt die Initialisierungsroutine in den Speicher und führt sie sofort aus.

Aus Spaß wurde schnell Ernst

Diese Einsprungroutine ist ein Nährboden für Viren, da sie bei jedem Start automatisch geladen und ausgeführt wird. Hängt sich ein Virus an diese Routine, gelangt er beim Start in den Speicher. Beim Booten einer befallenen Diskette passiert folgendes:

- Die modifizierte Routine reserviert Speicherplatz und kopiert sich selbst in den oberen RAM-Bereich.
- Der Reset-Vektor wird auf die im Speicher befindliche Virus-Routine umgeleitet.
- Anschließend führt das Virus-Programm den normalen Aufruf des Amiga-DOS aus.

Nun lauert der Virus. Niemand bemerkt ihn. Erst bei einem Reset wird er wieder aktiv. Statt der Reset-Routine springt das Betriebssystem in die Virus-Routine. Das Unheil nimmt seinen Lauf:

- Der Bootblock, der im internen Laufwerk befindlichen Diskette, wird wie üblich gelesen.
- Alle Anweisungen in der Startroutine werden ausgeführt — zum letzten Mal übrigens, wenn die Diskette nicht schreibgeschützt ist.
- Dann passiert es.
- Der gesamte Virus schreibt sich in die ersten beiden Sektoren der eingelegten Diskette.

Die schädigende Wirkung des Virus besteht in der Zerstörung des Bootblocks. Leider gibt es viele Programme, die dort schon wichtige Informationen enthalten. Spiele wie »Barbarian« oder »Terrapods« besitzen einen speziellen »Loader«. Solche Disketten werden durch einen Virus unbrauchbar gemacht.

Doch das ist nicht der einzige Effekt des SCA-Virus. Bei jedem 16. Reset meldet er sich auf dem Bildschirm. Scheinheilig, unschuldig erscheint eine Laufschrift, die über die Infektion berichtet.

Durch solche Publikumserfolge angestachelt, folgte auf dem Amiga eine Welle weiterer Viren:

Manche liegen nicht mehr im Bootblock, sondern verbergen sich in anderen Dateien. Das C-Directory ist ein beliebtes Versteck für allerlei Ungeziefer.

Doch es gibt weitere Brutstätten; in den Libraries, im Printer-Device, RAM-Handler-, Disk-Validator, einfach überall. Je heimtückischer der Programmierer, desto schlimmer die Folgen, desto schwerer läßt sich ein Virus aufspüren oder gar wieder kurieren. Ein Virus, der in der Lage ist, sich vor jedes Programm zu hängen, bei einem Aufruf ein neues Ziel sucht und eventuell sogar Hardware-Vektoren, Traps oder Systemroutinen in den Libraries manipuliert, ist nur schwer zu bekämpfen.

Eine derartige Spezies versteckt sich beispielsweise in einem modifizierten DIR-Befehl. Der Befehl läuft wie gewohnt, nur am Ende kopiert er einen Virus in den Speicher. Niemand merkt dies. Der Virus ist zu schnell.

Diese Abart mißbraucht zur Verbreitung den Disk-Validator, der immer aufgerufen wird, wenn Sie eine neue Diskette ins Laufwerk schieben. Was macht der Virus? Sobald eine Diskette nicht schreibgeschützt ist, kopiert er den unheilbringenden DIR-Befehl auf die eingelegte Diskette. Die negative Seite dieses Virus besteht glücklicherweise nur in der Verbreitung des falschen DIR-Befehls, verbunden mit der von Zeit zu Zeit auftauchenden Meldung, daß ein gewisser »König Samso« doch Herrscher aller Dinge sei.

Saver Copy

Zwei weitere Amiga-Viren machten von sich reden. Einer ist ein Abkömmling des SCA-Virus. Verbreitung und Versteck sind gleich. Das Krankheitsbild ist ungleich schwerer. Statt nur den Bootblock zu überschreiben, formatiert das Programm auch die Spur 40. Das gesamte Directory einer Diskette wird gelöscht.

Vielleicht haben Sie auch schon Bekanntschaft mit einem Virus gemacht, der zum totalen Systemabsturz führt. Selbst intensives Auslösen eines Resets hilft Ihnen nicht. Sie müssen Ihren Amiga ausschalten und neu starten.

Zumal die Folgen der Viren oft fatal sind, bleibt die Frage, ob es einen Schutz gibt?

Vor den bisher bekannten Erregern können Sie sich noch schützen. Der SCA-Virus und alle, die sich über den Bootblock verbreiten, sind durch den INSTALL-Befehl einfach auszulöschen. INSTALL sollte aber nur auf tatsächlich infizierte Disketten angewandt werden. Sie erkennen die Infektion am besten mit Hilfe eines Diskettenmonitors. Vergleichen Sie die Information in den ersten beiden Sektoren der Diskette mit denen einer nicht infizierten Workbench, oder einer mit INSTALL frisch behandelten Diskette. Meist befindet sich im Bootblock einer verseuchten Diskette sogar ein Hinweis auf den Programmierer des Virus. Beim SCA-Virus finden Sie in Block 1 die Meldung: »Something wonderful has happened. Your Amiga is alive«.

Um einen Virusbefall zu erkennen, können Sie Ihre Disketten auch mit dem »Boot-Boy« oder einem ähnlichen Programm mit einem Vorspann versehen. Diese Programme verwenden ebenfalls den Bootblock. Taucht beim Starten nicht mehr Ihr gewohntes Bild auf, können Sie davon ausgehen, daß sich ein Virus in Ihrem Amiga ausgebreitet hat.

Amiga 1000-Besitzer können sich sogar mit »Anti-Viren-Kickstarts« schützen. Sie zeigen Ihnen sofort an, wenn sich ein SCA-Abkömmling auf Ihrer Startdiskette befindet und verhindern dessen Aktivierung. Der SCA-Virus läßt sich sogar noch einfacher deaktivieren. Hierzu erhielten wir einen Tip direkt vom Programmierer:

- Sobald Sie sicher sind, daß sich der Virus im Amiga versteckt hält, lösen Sie einen Reset aus.
- Sofort nach dem Einlesen des Bootblocks drücken Sie die linke Maustaste.
- Der Bildschirm färbt sich kurz grün.
- Jetzt ist der Virus abgeschaltet. Er kann keine weitere Infektion bewirken.

Nicht immer gestaltet sich die Bekämpfung so einfach. Sie sollten auf jeden Fall Ihre Disketten nur mit Schreibschutz verwenden. So verhindern Sie, daß ein Virus sich auf die Disketten überträgt. Wenn Sie schreibend auf die Diskette zugreifen möchten, vergewissern Sie sich vorher, daß der Amiga virusfrei ist. Über Raubkopien oder Public Domain-Disketten, die ständig ihre Besitzer wechseln, verbreiten sich Viren. Besonders beim Kopieren von zweifelhaften Disketten — seien Sie auf der Hut. (Patrick Barile/ub)

News im Überblick

Desktop Video, das bedeutet: Videos produzieren mit Hilfe des Personal Computers. Der Amiga ist aufgrund seiner grafischen und akustischen Voraussetzungen hierzu prädestiniert.

Desktop Video

Die Anwendungsgebiete des Amiga im Bereich »Personal Video« reichen vom professionellen Titelgenerator über die Erstellung von Simulationen und Trickfilmen bis zur Bildmanipulation und Real-Trickmischung. Auf einer von Commodore einberufenen Pressekonferenz wurde neben der vielfältigen Grafiksoftware auch einiges an Hardware präsentiert.



Bild 1. Farbsplitter und VD3

Videocomp bietet für den Amiga drei Genlocks an:
— Prolock-HV ist laut Hersteller für den ambitionierten Videoamateur gedacht. Der Preis beträgt 1198 Mark.
— Den semiprofessionellen Kreis der Video-Anwender soll das für 1998 Mark erhältliche Genlock VCG-1 erschließen.
— VCG-2 ist ein professionelles Genlock für den Einsatz auf U-Matic und High-Band. Es kostet 3498 Mark.

Für alle Genlocks bietet Videocomp verschiedene Komplettlösungen an:
— Für den Amateur: Amiga 500, 512 KByte Speichererweiterung, Monitor 1084, 2. Laufwerk, Genlock HV, Deluxe Paint;
Komplettpreis: 4298 Mark
— Für den Semi-Profi: Amiga 2000, Monitor 1084, 2. Laufwerk, 2-MByte-Speichererweiterung, Genlock VCG-1, Deluxe Paint;
Komplettpreis: 6998 Mark
— Für den professionellen Producer: Amiga 2000, 2. Laufwerk, 20-MByte-Hard-Disk und SCSI-Controller, 4-MByte-Speichererweiterung, 68020/68881-Turbo-Board, High-Resolution-Monitor, Genlock VCG-2;
Komplettpreis: 17998 Mark



Der Amiga 2000 im kompletten Video-Studio Foto: Commodore

Merkens präsentierte einen Prototyp (Bild 1) des vollautomatischen Farbfilters für den Video-Digitizer VD 3. Dieser RGB-Farbsplitter wird voraussichtlich ab Dezember für 898 Mark erhältlich sein. Er erleichtert die Erfassung von Farbbildern und arbeitet zuverlässiger als die bisher verwendeten Farbscheiben.

Neben den auch schon auf der Systems vorgestellten Produkten (siehe AMIGA 12-'87, Seite 14) überraschte Merkens mit der Ankündigung eines Digitizers für den Amiga 2000. Die Steckkarte ist bereits in der Entwicklung und enthält neben einem angeblich wesentlich schnelleren und verbesserten Video-Digitizer einen integrierten RGB-Splitter. Als frühester Liefertermin wurde Anfang 1988 genannt. (ub)

Anbieter: Videocomp, Bergerstraße 193, 6000 Frankfurt/Main 60, Telefon (069) 467101
Merkens, Fuchstanzstraße 6A, 6231 Schwalbach, Telefon (06196) 3026

Gewinnerbilder

Die besten Bilder, die für den Malwettbewerb aus dem AMIGA-Magazin 10/87 von Lesern eingesandt wurden, sind jetzt auf zwei Disketten gesammelt erhältlich. Unter der Bestellnummer 48705 können Sie beide zusammen beim AMIGA-Programmservice zum Preis von 29,90 Mark bestellen. Lassen Sie es sich nicht entgehen, diese tollen Bilder in einer Slide-Show auf dem eigenen Bildschirm zu betrachten, denn dort sehen sie noch brillanter aus. (jk)

Markt & Technik Verlag AG, AMIGA-Programmservice, Hans-Pinsel-Str. 2, 8013 Haar, Tel. (089) 4613-0

Videoscape in PAL-Version

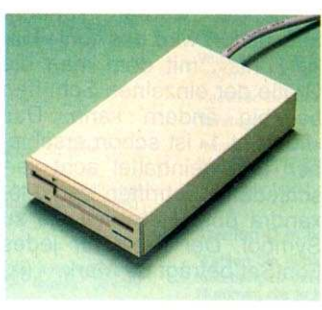
Softwareland wird den Vertrieb der neuen PAL-Version von Videoscape mit deutschem Handbuch übernehmen. Das komplette Paket soll zum Preis von 386 Mark noch vor Weihnachten angeboten werden. Außerdem ist auch Prowrite inzwischen mit deutschem Handbuch verfügbar. In Vorbereitung ist weiterhin »Go Amiga Text!«, das mit der schon erschienenen Dateiverwaltung »Go Amiga Datei!« zusammenarbeiten soll. (jk)

Softwareland, Franklinstr. 27, CH-8050 Zürich, Tel. (0041) 1-3-115959

Zusatzlaufwerk

Ein Zweitlaufwerk, das sich zum Anschluß an alle Amiga-Modelle eignet, kommt von FutureVision. Das in ein Blechgehäuse (Bild) eingebaute, abschaltbare NEC 1036A-Laufwerk kostet 329 Mark, mit durchgeschleiftem Bus 339 Mark. (dm)

FutureVision, Friedrich-Veith-Str. 21, 6128 Höchst/ODW., Tel. (06163) 1278 (ab 18.00)



Preiswertes Laufwerk

Der neue Lattice

Die neue Version des Lattice-C-Compilers ist jetzt knapp ein Jahr nach Erscheinen der Version 3.10 fertiggestellt. Beim Lattice-C-4.00 standen diesmal weniger die Libraries als vielmehr Compiler und Linker im Vordergrund der Verbesserungen. So wurden Codeoptimierungen eingebunden, die bereits am Objektmodul zu Verkürzungen von 5 bis 20 Prozent führen. Es können 16-Bit-Parameter auf dem Stack ohne lästiges »Umcasten« übergeben werden. Sogenannte »Function Prototypes« warnen vor Fehlern bei Funktionsaufrufen. Als Besonderheit kann der Programmierer nun wahlweise die Parameter an Bibliotheksfunktionen ohne den Umweg über den Stack direkt in Registern übergeben, wodurch Library-Aufrufe wesentlich beschleunigt werden. Eingebaute Funktionen zur String- und Speichermanipulation, die bereits vom Compiler bearbeitet werden und »Inline-Code« erzeugen, tragen noch mehr zur Beschleunigung bei.

BLINK, der zusammen mit den beiden Compiler-Durchläufen mit einem einzigen Kommando aufgerufen werden kann, ist in der Version 7.1 durch neues »Library-Indexing« sehr schnell geworden. So wurde beispielsweise ein aus 44 Einzelmodulen und zwei Libraries bestehendes Programm mit einer Gesamtlänge von über 100 KByte in nur 24 Sekunden gelinkt. Neben Overlays und der Möglichkeit zur interaktiven Symboldefinition während des Link-Vorgangs verfügt BLINK noch über einige weitere Optionen und verarbeitet trotz Erweiterung des Objekt-Formats noch immer Standardmodule.

Bemerkenswert ist, daß das Compiler-Paket, neben der umfangreichen und doch für Anfänger noch zu knappen Anleitung und vier Disketten, bereits die Include-Files der zu erwartenden Amiga-Betriebssystemüberarbeitung enthält. Alles in allem scheint auch der neue Lattice eine sehr lohnende Anschaffung zu sein. Käufer der alten Lattice-Version können vom Update-Service des Herstellers Gebrauch machen. (Ralph Babel/jk)

Konzept oder Personalwechsel?

Der bisher bei Commodore Deutschland verantwortliche Vertriebsdirektor für den Consumer-Bereich Helmut Jost hat sich neuen Aufgaben in einem anderen Unternehmen zugewandt. Die für den Vertrieb neu geschaffene Position des Gesamt-Vertriebsleiters übernimmt ab sofort der bisherige Vertriebsdirektor Rolf Wiehe. Commodore begründete das Ausscheiden und die damit verbundene Neuorganisation im Rahmen einer Straffung des Vertriebes und besseren Ausschöpfung des Marktpotentials. (jk)

Einstieg mit Amiga

Mit sechs Softwarepaketen hat die bisher nur aus dem Commodore 64-Geschäft bekannte Firma »Roßmöller Computer Tuning« einen Einstieg in die Amiga-Szene gestartet. Auf dem Sektor Spiele wurde dabei das Autorennen »Crazy Cars« vorgestellt. Mit hoher Geschwindigkeit fährt man verschiedene Super-Autos über bekannte Rennstrecken der Welt. Sehenswert ist auf jeden Fall die Endgrafik nach jedem Rennen. Als »Toolbox 1« ist eine Diskette betitelt, die außer mehreren IFF-Grafiken einen »Bootwriter« zum Erstellen von Vorspännen und einen Converter für Deluxe-Brushes in Sprite-Datas enthält.

Gleich zwei Programme sind auf den Sound-Sektor zugeschnitten. Der »Turbo Drummer« ermöglicht das Zusammenstellen verschiedener Drum-Sets, die aus digitalisierten Sounds bestehen und über die Tastatur gespielt werden können. Der »Effect-Creator« beinhaltet Tools zur Bearbeitung digitalisierter Sounds.

Weiterhin ist ein Lernprogramm für Englisch-Vokabeln erhältlich: »To Learn English«.

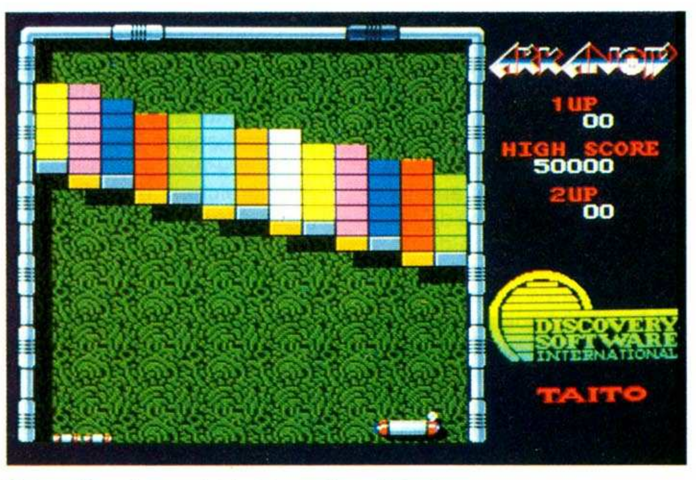
Auch auf das Gebiet Lagerverwaltung hat man sich mit dem Programm »Quick Facto« vorgewagt. Dieses Fakturierungsprogramm ist für Klein-Betriebe oder Vereine geplant und verwaltet deren Rechnungs- und Lagerwesen.

Die Programme Turbo Drummer und Quick Facto kosten je 149 Mark, während alle anderen für jeweils 99 Mark zu kaufen sind. Für den Effect Creator sowie den Turbo Drummer existieren auch schon Sound-Libraries, die je 49 Mark kosten. (jk)

Roßmöller Computer Tuning, Maxstraße 50-52, 5300 Bonn, Tel. 0228/650212

Echte Entdeckungen

Die amerikanische Softwarefirma Discovery, die sich bisher durch die hervorragenden Kopierprogramme der Marauder-Serie auf dem Amiga hervorgetan hat, veröffentlicht im nächsten Monat eine Umsetzung des bekannten Spielautomaten »Arkanoid« (Bild unten). Die Demo-Version überzeugte mit gut umgesetzter Grafik und sehr schnellen Bewegungen. Außerdem wird eine Reihe von Umsetzungen der bewährten Taschenrechner von Hewlett-Packard. Die DX-Serie umfaßt bereits den Scientific-Calculator DX-11C und für Programmierer den DX-16C mit vielen Umrechnungsarten. Damit kann man dann auf dem Amiga rechnen wie aus der Schule oder Universität gelernt. (jk)



Schneller Arkanoid jetzt auch auf Amiga

Farbige Disketten



Disketten in frohen Farben

Jetzt gibt es auch 3½-Zoll-Disketten in verschiedenen Farben (Bild oben). Die 2DD-Disketten werden von Bittendorf zum Preis von 3 bis 3,50 Mark angeboten, je nach Ausführung mit fünf bis zu 15 verschiedenen Farben. (jk)

Datentechnik Bittendorf, Stohrstr. 8, 6360 Friedberg, Tel. (06031) 61950

Vizawrite verbessert

Die Textverarbeitung Vizawrite Desktop liegt nun in der verbesserten Version 1.03 vor, die ab sofort ausgeliefert wird. Besitzer der Version 1.01 oder 1.02 können nach Angaben des Anbieters einen kostenlosen Update-Service in Anspruch nehmen. Dazu muß nur die Originaldiskette mit einem frankierten Rückumschlag und der Bitte um Update an DTM eingeschickt werden.

Für Besitzer eines NTSC-Amiga ist Vizawrite Desktop nun auch angepaßt lieferbar. Der Preis für das NTSC-Produkt: 198 Mark. (dm)

DTM, Poststr. 25, 6200 Wiesbaden, Tel. (06121) 56 0084

Klassische Schriften

Wer bekannte Schriftarten für Textprogramme auf dem Amiga sucht, für den gibt es jetzt die Sammlung »FontSets« von Gold Disk. Darin sind Schriften enthalten, die in jedem Amiga-Textprogramm verwendet werden können. Mitgeliefert wird das Tool »Make Fonts«, mit dem man die Größe der einzelnen Schriften beliebig ändern kann. Das »FontSet 1« ist schon erschienen und beinhaltet acht verschiedene Schriften von Alexandra über Helvetica bis zu Symbol. Der Preis für jedes FontSet beträgt 89 Mark. (jk)

Soltau Electronics, Esplanade 39, 2000 Hamburg 36, Tel. 040/340445

Deutschlands Programmierer

Vermarktung, Programmierung und Support waren Themen auf dem ersten selbstinitiierten Treffen der Amiga-Programmierer. Der neue, daraus resultierende Erfahrungsaustausch soll nicht nur helfen, Software zu verbessern, sondern auch, sich besser über die Möglichkeiten des Verkaufs eigener Produkte zu informieren.

Thema Nummer 1 für die Anwesenden war natürlich das Programmieren. Man tauschte Erfahrungen und Problemlösungen aus.

Das Gesprächsthema mit den hitzigsten Diskussionen war der Verkauf eigener Software an Softwareverleger. In Zukunft will man sich bei der Durchschaubarkeit von Verträgen gegenseitig helfen und teilweise auch schon gemeinsam verhandeln. Schlechte Zeiten für die Softwarefirmen kommen trotzdem nicht, denn man hat realistische Vorstellungen vom Preis. Eines war allerdings allen klar: Man sollte sich bei allen Softwarehäusern nach den Konditionen erkundigen und vielleicht eher Überredungskünstler an den Verhandlungstisch lassen.

Oliver Czesla stellte mit »Phalanx II« sein viertes professionelles Amiga-Spiel vor. Phalanx II ist ein Spiel vom Typ Uridium und wird in Grafik und Sound dem Amiga gerecht. Wann es erscheinen wird, stand bei Redaktionsschluß noch nicht fest.

Die Gebrüder Brenner demonstrierten das Actionspiel »Alcazzar«. Eines der interessantesten Werke stammt von Thomas Klinger und ist sein Erstling. Unter dem Projektnamen »Space Star« entsteht ein Ballerspiel, das auch auf Spielautomaten seinesgleichen sucht. Drei Autoren hatten sich die Umsetzung von Spielautomaten des Typs »Nemesis« und »Salamander« vorgenommen. Eines dieser drei Actionspiele konnte mit Grafik aufwarten, die man so noch nicht gesehen hat; plastische Explosionen und schimmernde Gläser flogen da über den Bildschirm. »X« ist der Arbeitstitel des Programmes, das frühestens Mitte nächsten Jahres erscheint. »Neutralizer«, zweites Spiel des Genres, von einem Kölner Programmierteam, steckt ebenfalls mitten in der Entwicklung und konnte mit erstaunlichen Grafik- und Sound-

effekten aufwarten. »Sarcophaser«, der dritte Vertreter, orientiert sich im spielerischen Element mehr an Nemesis, in der Grafik mehr an Salamander und natürlich, wie alle drei, an vielen Neuheiten. Ein Geschicklichkeitsspiel mit Namen »Emetic Skimmer« der Gruppe namens »Hack & Slay« ist bereits fertig. Ein Fantasy-Spiel im Stil der Klassiker »Wizardry« ist fast vollendet: »Witchcraft« ist ein Rollenspiel, in dem man ausnahmsweise nicht gegen einen bösen Zauberer kämpfen muß.

Bemerkenswert war — Ausnahmen bestätigen die Regel — daß Software, die unter Zusammenarbeit in Programmiergruppen entstand, in der Qualität besser war als Soloproduktionen. Der Trend zu effektiver Gruppenarbeit ist deutlich erkennbar. Soundroutinen-Autoren, Grafiker, sogar Testspieler sind im Konzept inbegriffen. Jetzt, nach dem Treffen, wird der nächste Schritt getan: Eine Verbindung der Gruppen untereinander soll den Tausch von Routinen, Tools und Erfahrungen vereinfachen. Weitere Treffen sind geplant. Warten wir ab, ob aus der Idee der neuen Einigkeit eine dauerhafte Verbindung wird. (M. Kohlen/jk)

Preiswerte Festplatte für den A2000



Filecard zum Einstecken für A2000/PC-Karten-Besitzer

DTM bietet zum Preis von 748 Mark die »Business Card 21« (Bild) von Tandon (Filecard inklusive Controller) an, die sich zum Einstecken in einen der PC-Slots des Amiga 2000 eignet (läuft nur mit PC-/AT-Karte). Die 76 Millisekunden schnelle Platte hat eine formatierte Gesamtkapazität von etwa 20 MByte. Sie benötigt keine Stromzufuhr über externe Kabel, da sie die benötigte Energie dem Bussystem entnimmt. Leider besitzt die Filecard keinen Anschluß für eine Kontroll-LED. Mit den geeigneten Programmen soll die Platte sowohl vom PC aus wie auch vom Amiga (als »jh0«) benutzt werden können. (dm)

DTM, Poststr. 25, 6200 Wiesbaden, Tel. (06121) 560084

Inserenten

* Anzeige mit aufgeklebter Postkarte

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Geschäftsführender Chefredakteur: Michael Scharfenberger

Chefredakteur: Albert Absmeier (aa)
Stellvertr. Chefredakteur: Georg Klinge (gk)
Redaktion: Peter Aurich (pa), René Beaupoil (rb), Ulrich Brieden (ub), Jörg Kähler (jk), Dieter Mayer (dm)

Hotline: Gerd Donaubauer (640), Monika Welzel (640)

Redaktionsassistenz: Monika Lewandowski (222), Andrea Kaltenhauser (202)

Fotografie: Jens Jancke, Claudia Kränzle

Titelgestaltung: Heinz Rauner, Grafik Design, Werner Nienstedt

Layout: Leo Eder (Ltg.), Rolf Raß (Cheflayouter), Dagmar Berninger, Willi Gründl

Auslandsrepräsentation:
Schweiz: Markt & Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042-415656, Telex: 862329 mut ch
USA: M & T Publishing, Inc: 501 Galveston Drive, Redwood City, CA 94063; Tel. (415) 366-3600, Telex 752-351

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß Markt & Technik Verlag Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Produktionsleitung: Klaus Buck

Anzeigenverkaufsleitung: Ralph Peter Rauchfuss (126)

Anzeigenleitung: Alicia Clees (313)

Anzeigenverwaltung und Disposition: Patricia Schiede (172), Monika Burseg (147)

Anzeigenformate: 1/1-Seite ist 266 Millimeter hoch und 185 Millimeter breit (3 Spalten à 58 mm oder 4 Spalten à 43 Millimeter). Vollformat 297 x 210 Millimeter. Beilagen und Beihefter siehe Anzeigenpreisliste.

Anzeigen-Auslandsvertretungen:
England: F. A. Smyth & Associates Limited 23a, Aylmer Parade, London, N2 OPQ, Telefon: 0044/1/3405058, Telefax: 0044/1/3419602
Taiwan: Third Wave Publishing Corp. 1 — 4 Fl. 977 Min Shen E. Road, Taipei 10581, Taiwan, R.O.C., Telefon: 00886/2/7630052, Telefax: 00886/2/7658767, Telex: 078529335

Vertriebsleiter: Helmut Grünfeldt (189)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebsgesellschaft mbH, Hauptstätterstraße 96, 7000 Stuttgart 1, Telefon (0711) 6483-0

Bezugsmöglichkeiten: Leser-Service: Telefon 089/4613-249. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen. Das Abonnement kann jederzeit zum Ende des bezahlten Zeitraums gekündigt werden.

Bezugspreise: Das Einzelheft kostet DM 7,—. Der Abonnementspreis beträgt im Inland DM 79,— pro Jahr für 12 Ausgaben. Darin enthalten sind die gesetzliche Mehrwertsteuer und die Zustellgebühren. Der Abonnementspreis erhöht sich um DM 18,— für die Zustellung im Ausland, für die Luftpostzustellung in Ländergruppe 1 (z. B. USA) um DM 38,—, in Ländergruppe 2 (z. B. Hongkong) um DM 50,—, in Ländergruppe 3 (z. B. Australien) um DM 68,—.

Druck: R. Oldenbourg GmbH, Hürderstr. 4, 8011 Kirchheim

Verantwortlich: Für redaktionellen Teil: Albert Absmeier. Für Anzeigen: Alicia Clees.

Redaktions-Direktor: Michael M. Pauly

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 522052

Telefon-Durchwahl im Verlag:

Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089-4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.

Zufallsgenerator

Ich möchte in Maschinen-Code eine Routine schreiben, die Zufallspunkte auf einem Grafik-Screen setzt. Meine Frage: Wie kann ich in Assembler Zufallszahlen produzieren? INGMAR REYER

Wersi-Board

Seit meinen 64'er Zeiten habe ich ein Wersi-Board, mit dem ich auf dem C 64 sehr gute Musik machen konnte. Die Frage ist, wie kann ich das Wersi-Board an den Amiga anschließen?

Natürlich ist mir klar, daß der Expansionportstecker des Wersi-Boards nicht an den Port des Amiga 500 paßt. Ich kann mir denken, daß es möglich ist, ein Interface für den Amiga zu basteln oder zu kaufen. Die passende Software kann ich selbst schreiben. Wer kann mir helfen?

SERGIO LENCINAS

Jugend forscht

Für ein neues Projekt im Wettbewerb »Jugend forscht« suchen wir, die Arbeitsgemeinschaft Astronomie, die 1986 Landessieger des Wettbewerbs in NRW war, dringend eine Schaltung für einen A-/D-Wandler, mit der wir einen Amiga 2000 als Registriervoltmeter einsetzen können. Unsere neue Arbeit beschäftigt sich mit der Digitalisierung von Bildern und deren rechnergestützten Verarbeitung.

JÖRG SÄNGER i.A.
Arbeitsgemeinschaft Astronomie

Zweitlaufwerke

Ich habe mir vor kurzem ein NEC-1036 A-Laufwerk als Zweitlaufwerk zum Amiga 2000 gekauft. Nun stehe ich vor folgendem Problem:

Wie kann ich das Laufwerk umbauen, damit es die Diskettenwechsel selbständig (wie ein eingebautes Original-Laufwerk) erkennt?

Vielleicht hat ein Leser das Problem schon gelöst und kann mir helfen.

HEIKO JAHN

Sprites und Compiler

Ich versuche seit geraumer Zeit, mit »Manx Aztec-C« und der Hilfe des »ROM Kernal Manuals« »VSprites« zu programmieren. Mit den Simple Sprites habe ich keinerlei Probleme mehr, aber irgend etwas scheint im ROM Kernal Manual in bezug auf VSprites falsch zu sein. Da ich noch nicht so lange C programmiere, habe ich nicht die Erfahrung, den Fehler selbst zu finden. Wer hilft mir?

ARMIN NUSER
Ausgabe 8/9



Unsere Redakteure Ulrich Brieden und Jörg Kähler wählen gemeinsam die Beiträge für das Leserforum aus

Zum Leserbrief von Armin Nuser in Heft 8/9:

Die Probleme, die auftreten, könnten daran liegen, daß nicht mit der Option "+l" compiliert wurde, da einige Systemroutinen (die mit Lattice-C geschrieben wurden) abstürzen, wenn Integer keine 32 Bit lang sind. AXEL PETERS

Arbeitshilfe

Anscheinend ist für das vernünftige Arbeiten mit dem CLI des Amiga 500 ein zweites, externes Laufwerk nötig, eins für die Workbench-Diskette, das andere für die zu bearbeitende Diskette. Denn der Computer verlangt bei jedem CLI-Befehl die WB-Disk, falls eine andere Disk eingelegt wurde, wobei dann der Befehl auch nur auf diese bezogen wird. Ich kann also bei einem Laufwerk im CLI nur mit der WB-Disk arbeiten, was wenig sinnvoll ist. Eine von mir festgestellte Ausnahme ist der Befehl FORMAT, bei dem nach der WB-Disk die zu formatierende Disk verlangt wird. Im CLI-Kurs konnte ich zu diesem Problem keinen Hinweis finden.

KLAUS LEIN

Sie brauchen nicht unbedingt ein zweites Laufwerk. Es gibt schon zahlreiche Tips, um auch mit einem Laufwerk vernünftig mit dem CLI zu arbeiten. Die häufigste Methode ist das Anlegen einer RAM-Disk. In diese können Sie die benötigten CLI-Befehle kopieren und mittels PATH den Amiga anweisen, die Befehle erst in der RAM-Disk zu suchen (siehe Tips und Tricks). Ein weiterer Trick ist es, wenn Sie zum Beispiel beim DIR-Befehl den Namen der Diskette, von der Sie das Verzeichnis wünschen, mitangeben:

```
DIR DISKNAME:
```

Zuerst liest der Amiga den Befehl DIR von der eingelegten System-Diskette, dann fordert er Sie auf, die betreffende Diskette einzulegen.

Noch raffinierter ist es, wenn Sie nach dem DIR ein Fragezeichen eingeben.

```
DIR ?
```

Jetzt liest der Amiga zunächst den Befehl und wartet auf eine Eingabe. Legen Sie nun Ihre Diskette ein und drücken <RETURN>. Schon erscheint das Verzeichnis der Diskette. Auf dieselbe Art und Weise können Sie die anderen Befehle des CLI verwenden. (ub)

5¼-Zoll-Laufwerke

Wie ist das mit den 5¼-Zoll-Laufwerken? Gibt es überhaupt welche, die man ohne Probleme an den Amiga anschließen kann?

Wie arbeiten sie mit Kopierprogrammen oder ähnlichen Befehlen?

Was kosten Sie?

INGMAR BODE

Viele Fragen betreffen den Betrieb von 5¼-Zoll-Laufwerken am Amiga. Sie werden von zahlreichen Händlern angeboten. Der Vorteil solcher Laufwerke? Der häufigste Grund sind die niedrigen Preise der 5¼-Zoll-Disketten im Vergleich zu 3½-Zoll. Zudem erlauben einige dieser Laufwerke das Arbeiten im MS-DOS-Format mit dem MS-DOS-Emulator beziehungsweise mit dem Sidecar (PC-Karte beim Amiga 2000).

Natürlich taucht die Frage auf, ob diese Laufwerke vollständig kompatibel zu den externen 3½-Zoll-Laufwerken sind. Im Prinzip schon. Zumindest die neuerdings angebotenen 5¼-Zoll-Laufwerke sind genau wie ein 3½-Zoll-Laufwerk zu gebrauchen. Der Anschluß erfolgt mittels Kabel am Diskport. Die Laufwerke werden automatisch erkannt und arbeiten mit allen Befehlen:

— DISKCOPY
— FORMAT
— INSTALL

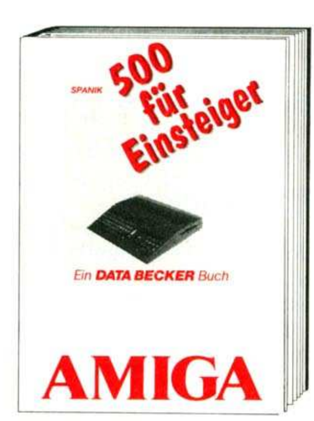
Manche der angebotenen Laufwerke bieten sogar den zusätzlichen Service einer Umschaltung zwischen 80 und 40 Tracks (MS-DOS).

Bei manchen älteren Laufwerken erkennt der Amiga den Wechsel der Disketten nicht (siehe oben). Hierzu ist der Befehl DISKCHANGE vorgesehen. Die neueren Modelle sind komfortabler. Achten Sie diesbezüglich auf die Angaben des Anbieters. Ein Nachteil ist, daß sich Programme vom externen 5¼-Zoll-Laufwerk nicht ohne

AMIGA BUCHHITS



Vom C64 zum Amiga? Nichts leichter als das. Mit dem Aufsteigerbuch. Neben einem schnellen Einstieg in Workbench, CLI und Multitasking findet der Aufsteiger hier einen Einblick in die wichtigsten Sprachen des Amiga – alles immer im Vergleich zu Ihrem alten Rechner. Für den eigentlichen Daten-Transfer von Texten, Bildern, Dateien und BASIC-Programmen enthält das Buch fertige Konvertierungsprogramme. Einen leichteren Umstieg werden Sie nicht finden.
Amiga Aufsteigerbuch.
Hardcover, 320 Seiten, DM 39,–



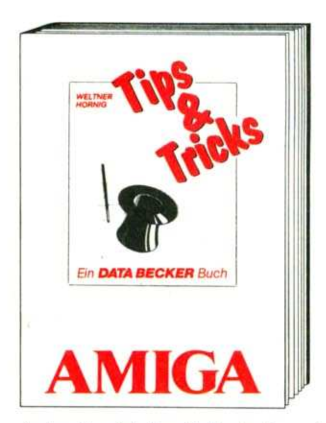
Wählen Sie gleich den richtigen Einstieg zu Ihrem Amiga 500. Denn das Handbuch läßt Sie dabei völlig allein. Versuchen Sie es lieber gleich mit Amiga 500 für Einsteiger. Hier heißt es: anschließen und loslegen. Verständlich für jedermann zeigt Ihnen dieses Buch: Workbench, AmigaBASIC, CLI und AmigaDOS. Locker aufbereitet bietet es Ihnen alles Wissenswerte. Bis hin zu den beim Amiga 500 mitgelieferten Zusatzprogrammen.
Amiga 500 für Einsteiger
343 Seiten, DM 39,–



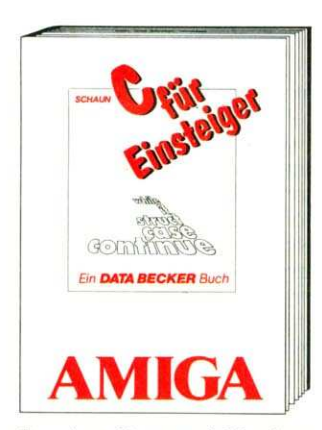
Läßt das Handbuch Sie auch in so manchen Dingen allein, das große Amiga 2000 Buch nicht. Hier finden Sie eine umfassende Einführung in die Arbeit mit Ihrem neuen Rechner – und mehr als das. Sind Sie erst einmal mit dem Amiga 2000 „per Du“, zeigen Ihnen die Autoren, was einen Amiga-Profi ausmacht: Kickstart im RAM, PC-Audioausgänge, erste Hilfe bei Harddisk-Abstürzen, Laufwerkeinbau in den Amiga 2000 und, und, und. Sollten Sie also noch Fragen zu Ihrem Rechner haben, hier finden Sie die Antworten.
Das große Amiga 2000 Buch
Hardcover, 684 Seiten, DM 59,–



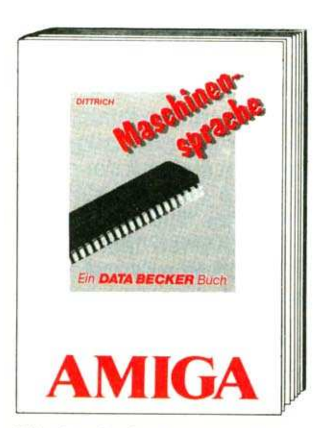
Das erfolgreiche Buch zu Amiga-BASIC – jetzt in Neuauflage! Erweitert um Kickstart 1.2, neuer Workbench und Amiga 500 & 2000. Mit allem, was BASIC-Programmierern Spaß macht: Grafik und Sound, Laden und Speichern von Graficraft-Bildern in BASIC-Programme, sequentielle und relative Dateien, Busineß-Grafik, Computeranimation, Windows, Umgang mit IFF-Bildern, Sprachausgabe und, und, und. Das Buch für Einsteiger, Aufsteiger und Profis.
AmigaBASIC
Hardcover, 774 Seiten, DM 59,–



Amiga Tips & Tricks. Ein Buch, das voller Überraschungen steckt: 64 Farben gleichzeitig auf dem Amiga. Von BASIC aus Zugriff auf die Libraries. Benutzung verschiedener Zeichensätze in BASIC. Sinnvoller Einsatz von Windows, Screens und Menüs. Tips zu einzelnen Grafikbefehlen, Programm- und AmigaDOS-Routinen! Greifen Sie in die Trickkiste, und schon sind Dinge möglich, die man gar nicht gedacht hätte.
Amiga Tips & Tricks
Hardcover, 364 Seiten, DM 49,–



C an einem Wochenende? Durchaus möglich! Mit C für Einsteiger. Ein Einführungskurs, der Ihnen schnell und einfach die wichtigsten Grundlagen dieser Sprache vermittelt. Vom ersten Programm bis hin zu den Routinen in den Bibliotheken. Mit dem gesamten Sprachumfang und den besonderen Features von C. Zahlreiche Tips & Tricks zur Programmierung und eine Beschreibung der beiden Compiler Lattice C und Aztek runden das Ganze ab.
Amiga C für Einsteiger
293 Seiten, DM 39,–



Schreiben Sie Ihre Programme in Maschinensprache – und Sie werden sehen, wie schnell ein Amiga sein kann. Das nötige Know-how liefert Ihnen dieses Buch: Grundlagen des 68000, das Amiga-Betriebssystem, Druckeransteuerung, Diskettenoperationen, Sprachausgabe, Windows, Screens, Register, Pull-Down-Menüs ... Und damit Sie auch gleich praktisch arbeiten können, werden die wichtigsten Assembler vorgestellt.
Amiga Maschinensprache
Hardcover, 288 Seiten
DM 49,–

Super

Regelmäßig in der DATA WELT: Amiga Window – das Forum für den engagierten Amiga-Anwender. Mit kreativen Projekten, Interviews, Software-Tests und wichtigen News. Und: Auch was sich sonst so in der Computerszene tut, erfährt der Amiga-Anwender. Die DATA WELT – das aktuelle Computermagazin. Monat für Monat überall da, wo es Zeitschriften gibt.

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

BESTELL-COUPON

Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:

zzgl. DM 5,– Versandkosten
unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Name
Straße
Ort

weiteres booten lassen. Es gibt zwar einige Tricks hierzu (siehe Tips und Tricks Ausgabe 12), aber diese funktionieren nicht immer. Natürlich haben Sie dieselben Schwierigkeiten, wenn Sie ein Programm von einem externen 3½-Zoll-Laufwerk starten wollen. Wenn in diesem Fall aber alle Versuche fehlschlagen, können Sie die Diskette wenigstens im internen Laufwerk »df0:« benutzen.

Im Zweifelsfall sollten Sie die Frage der Kompatibilität mit dem Händler vorab klären. Die Preise der 5¼-Zoll-Laufwerke können Sie den Anzeigen in der AMIGA entnehmen. Grundsätzlich liegen sie zur Zeit etwas höher als die der 3½-Zoll-Laufwerke. (ub)

GFA-Basic

Da ich zur Zeit (etwas enttäuschter) Besitzer eines Atari ST bin, habe ich natürlich auch Bekanntschaft mit dem wirklich guten GFA-Basic gemacht. Nun möchte ich gerne wissen, ob man bei GFA daran denkt, dieses auf dem Amiga anzubieten?

INGMAR BODE

An einer Umsetzung des GFA-Basic auf dem Amiga wird bereits fieberhaft gearbeitet. Wann es fertig ist und in den Handel kommt, steht noch nicht fest. Erste Versionen werden hoffentlich noch im ersten Quartal 1988 erhältlich sein. (ub)

RAM-Erweiterung ausschalten

Wie kann man die A501-RAM-Erweiterung hardwaremäßig abschalten, ohne die Uhr mit auszuschalten? Der Eingriff sollte an der Erweiterung, nicht am Computer erfolgen. Darf diese Abschaltung auch durchgeführt werden, wenn der Computer im Betrieb ist? Was bedeutet das File »SlowMemLast« auf der Version 33.56 der Workbench?

ALFRED SCHAMBURGER

— Eine Anleitung zum Umbau der Speichererweiterung finden Sie in dieser Ausgabe in der Rubrik »Tips und Tricks«. Mit dieser können und sollten Sie die Erweiterung bei ausgeschaltetem Amiga umschalten.

— »SlowMemLast« veranlaßt den Amiga, Programme und Daten zunächst im Fast-Memory abzulegen. Das Chip-Memory wird nur belegt, wenn eine Funktion wie »AllocMem« ausdrücklich Chip-Memory verlangt. Mit dieser Option können Sie Programme im schnellen Fast-RAM ablaufen lassen. Sie müssen aber daran denken, daß die Spezialchips des Amiga nur auf das Chip-Memory zugreifen können. Die Wirkung von »SlowMemLast« wird durch einen zweiten Aufruf des Programms wieder aufgehoben. (ub)

Schaltpläne

Kann man Amiga-Schaltplatinen oder -Bestückungspläne und Spezial-Chips kaufen (Gary, Fat Agnus, Denis, Paula etc.)? Wenn ja: Wo?

THORSTEN BÖGE

Die Schaltpläne des Amiga können Sie unter anderem über die folgende Adresse beziehen:
Schaltungsdienst Lange
Postfach 470653
Moriner Allee 30
D-1000 Berlin 47

Die Spezialchips des Amiga werden von Commodore hergestellt. Sie sollten sich deshalb an Commodore wenden, wenn Sie die Chips kaufen möchten. (ub)

Druckerprobleme

Ich besitze einen Commodore MPS 1000. Das Drucken von Text funktioniert einwandfrei. Bei der Ausgabe von Grafik gibt es allerdings ein Problem. Die letzte Pixelreihe der oberen Zeile überschneidet sich mit der ersten Pixelreihe der nächsten Zeile, was das Bild stark »streifig« erscheinen läßt. Dieser Effekt tritt immer auf, wenn Grafik gedruckt wird, so auch mit dem Notepad der Workbench und jedem Hardcopyprogramm. Was kann man dagegen tun?

HARALD THOIS
Ausgabe 10

Ich hatte mit meinem Star NL-10 die gleichen Probleme, denn Commodore hat in seinem Benutzerhandbuch empfohlen, den Epson-Druckertreiber für den NL-10 zu verwenden.

Nach langem Ausprobieren habe ich herausgefunden, daß man dieses Manko einfach beheben kann, indem man einen anderen Druckertreiber verwendet. Mit dem MPS 2000-Druckertreiber hatte ich die besten Erfolge. Es entstehen sehr gute Hardcopies. Zusätzlich hat man die Möglichkeit, mit dem Treshold-Menü die Punktdichte zu verändern. Näheres steht im Heft »Benutzerhinweise für Basispaket, Seite 78«. Man muß lediglich vermeiden, Treshold größer als 6 zu wählen, da sonst eine zu hohe Auflösung eingestellt wird, die der NL-10 nicht besitzt.

Ich könnte mir vorstellen, daß der Druckertreiber für den MPS 2000 dieselben positiven Eigenschaften bei Herrn Thois bewirkt wie bei mir.

MARKUS HEIDE

Bibliotheken

Ich bin stolzer Besitzer eines Amiga 500. Da mir der Basic-Interpreter des A500 zu langsam ist und ich bereits die Maschinensprache des C 64 beherrsche, kaufte ich mir vor kurzer Zeit einen Assembler und ein Buch über Maschinensprache.

Hier bin ich auf ein Problem gestoßen. In dem Buch wird oft geschrieben, daß man eine Bibliothek (zum Beispiel die »dos.library«) laden muß. Ich habe aber beim besten Willen keine solche Bibliothek auf der Workbench-Diskette gefunden.

Leider weiß ich nicht, wo diese Routinen stehen und wie man sie verwenden kann. Wer kann mir helfen?

DIRK FAUST

In den Bibliotheken befinden sich die System-Routinen des Amiga. Es gibt mehrere solcher »Libraries«. Im ROM-Kernal Manual finden Sie Informationen, welche Routinen in welcher Bibliothek stehen. Zahlreiche der Bibliotheken, zum Beispiel die genannte »dos.library«, befinden sich im Kickstart-ROM beziehungsweise auf der Kickstart-Diskette des Amiga 1000. Diese residenten Libraries müssen also nicht von der Diskette geladen werden. Einige andere können Sie im Ordner »Libs« auf der Workbench-Diskette finden. Um die Routinen einer Bibliothek in eigenen Programmen zu nutzen, müssen Sie die Bibliothek öffnen. Das Betriebssystem prüft zunächst, ob die Bibliothek sich bereits im Speicher befindet oder ob sie erst geladen werden muß. Ist eine Library ordnungsgemäß geöffnet, können Sie alle Routinen mittels »JSR« anspringen. Einige Beispiele dieser Technik und genauere Erklärung finden Sie in der AMIGA-Rubrik »Programmieren«, zum Beispiel in Ausgabe 11 auf Seite 92. (ub)

Startfragen

1. Was ist »booten«?
2. Wie kann der Amiga 512 KByte adressieren, und wie viele Bänke besitzt er?
3. Wie kann man Programme mit Icons speichern, und diese überhaupt definieren?

»Booten« heißt den Amiga mit einer Systemdiskette starten. Nach Einschalten des Amiga fordert dieser Sie auf, die Workbench-Diskette einzulegen. Dies ist Ihre Startdiskette. Legen Sie die Diskette ein. Mit ihr »bootet« der Amiga. Jetzt liest der Amiga die ersten beiden Spuren der Diskette, den sogenannten Bootblock. An einer Kennung registriert der Computer, ob es sich bei der eingelegten Diskette um eine gültige Startdiskette im Amiga-DOS-Format handelt. Im Bootblock befindet sich zusätzlich noch ein Assemblerprogramm, das der Amiga beim Starten ausführt; die DOS-Initialisierungsroutine. Sie öffnet die »dos.library« und übergibt die Kontrolle an Amiga-DOS. Am Ende des Vorgangs steht die Ausführung der »Startup-Sequence«.

— Der 68000-Prozessor im Amiga verfügt über einen 24 Bit breiten Adreßbus. Er kann 16 MByte direkt ansprechen. Das heißt: Kein Banking, keine umständliche Hardware, sondern alle Daten und Programme sind immer direkt zugänglich.

— Icons können Sie mit dem Icon-Editor, der sich auf der Workbench befindet, erstellen. Zu einem beliebigen Programm, auch wenn es bisher noch kein Icon besitzt, können Sie ein eigenes Icon malen. Sie müssen das mit »IconEd« gezeichnete Bild nur unter dem Namen Ihres Programms abspeichern. Programm und Icon müssen sich im gleichen Dateiverzeichnis befinden. Der einzige Unterschied zwischen dem Namen des Programms und des Icons ist das angehängte Kürzel ».info«. Dieses wird von »IconEd« automatisch an den Namen angehängt. Sie können einem Programm auch das Pictogramm eines anderen Programms zuordnen. Mit dem CLI-Befehl COPY ist dies recht einfach. Kopieren Sie die Info-Dateien:

```
COPY Prog1.info TO
Neuesprog.info
```

Wenn Sie Basic-Programme speichern, versieht Amiga-Basic das Programm automatisch mit einem Icon. (ub)



Wie sage ich es dem Amiga

Wozu gibt es so viele Programmiersprachen? Weil jede Sprache ganz spezielle Eigenschaften besitzt. Lernen Sie die Sprachen kennen, mit denen Sie auf dem Amiga programmieren können.

Programmiersprachen sind das Bindeglied zwischen Mensch und Computer:
— Sie stellen dem Computer eine Aufgabe in einer bekannten Programmiersprache.
— Er übersetzt das Programm in die interne Maschinensprache, in Befehle, die der Prozessor in seinem Innern versteht — und führt schließlich Ihre Anweisungen aus.

Zur Übersetzung des Programms benötigt der Computer Software; einen Interpreter oder einen Compiler für die eingesetzte Sprache.

Ein Interpreter liest ein Programm zeilenweise, übersetzt die Zeile und führt sie aus. Ein Vorteil ist, daß Sie ein Programm schreiben und direkt ausprobieren können. Alle Fehler werden beim Testen sofort erkannt und können von Ihnen korrigiert werden. Sie sind aber immer auf den Interpreter angewiesen, der einen Teil des Speichers belegt.

Ein Compiler übersetzt das gesamte Programm in Maschinencode. Den »Quellcode«, der übersetzt werden soll, schreiben Sie mit Hilfe eines Texteditors, beispielsweise dem ED des CLI. Die Compilierung liefert ein schnelles und eigenständiges Maschinenprogramm. Sie müssen Ihren Quellcode allerdings nach jeder Änderung erneut übersetzen. Der Aufwand bei der Entwicklung eines Programms ist dementsprechend groß, zumal Compiler nicht so einfach zu bedienen sind wie Interpreter.

Manche Compiler liefern Ihnen nicht direkt das lauffähige Programm, sondern erst den »Objektcode«. Sie können in solch einem Fall ein großes Projekt in kleinere Module zerlegen und jedes einzeln compilieren. Sie können sogar vorgefertigte Routinen aus Bibliotheken in Ihre Programme einbinden. Der »Linker« verbindet anschließend alle erforderlichen Objektdateien zu einem ausführbaren Programm.

Mehrere Compiler oder Interpreter für unterschiedliche Sprachen stehen Ihnen zur Kommunikation mit dem Computer zur Verfügung.

Das ABC

Assembler, Basic und C sind die bekanntesten Sprachen für den Amiga. Doch es gibt mehr. Wozu so viele Sprachen? Würde nicht eine reichen?

Ein Grund liegt an den unterschiedlichen technischen Voraussetzungen der Computer



und den Anforderungen, die Anwender an ihren Computer stellen. Der eine möchte mit hoher Genauigkeit rechnen, das kostet eventuell Zeit und Speicherplatz. Ein anderer benötigt schnelle Routinen, hierzu braucht er eine möglichst maschinennahe Sprache.

Interessieren Sie sich für die Programmierung kommerzieller Anwendungen, für Spiele oder Systemprogrammierung?

Wenn Sie programmieren möchten und die Schwerpunkte Ihrer Programme kennen, können Sie die Sprache auswählen, die Ihren Anforderungen am weitesten entgegenkommt. Bevor Sie eine lernen, schauen Sie sich an, welche anderen es für den Amiga bereits gibt.

Basic – die Basis

Beginnen wir unsere Exkursion durch die Welt der Programmiersprachen mit Basic. Das ist die im Bereich der Heimcomputer verbreitetste Sprache. Jeder Amiga-Besitzer verfügt über eine Basic-Version: das Amiga-Basic.

Basic: »Beginners All Purpose Symbolic Instruction Code«.

Die Sprache soll für Anfänger leicht zu erlernen und für alle Computeranwendungen einsetzbar sein.

Diesen Anforderungen wird Basic auch im wesentlichen gerecht. Sie können in Basic sowohl Anwendungen wie Textverarbeitungen oder Dateiverwaltungen programmieren, als auch Spiele oder Grafikprogramme.

Speziell das Amiga-Basic unterscheidet sich von vielen älteren »Basic-Dialekten«. Es ist weitaus komfortabler. Vor allem erlaubt es eine strukturierte Programmierung. Strukturiert programmieren heißt auch übersichtlich programmieren. Jeder soll diese Programme lesen und nachvollziehen können. Keine undurchschaubaren Sprünge zu irgendwelchen Zeilennummern oder sonstige Unarten, die selbst der Urheber eines solches »Spaghetti-Codes« nach kurzer Zeit nicht mehr versteht.

Amiga-Basic kommt ohne Zeilennummern aus. Als Sprungadressen werden Sprungmarken, »Label«, verwendet. Auf unübersichtliche GOTO-Sprünge kann verzichtet werden. Eine strukturierte Sprache erlaubt das blockweise Abarbeiten eines Programms. Amiga-Basic ermöglicht dies beispielsweise durch die leistungsfähigen IF-THEN-ELSE-ENDIF-Befehle.

Programmschleifen können Sie sowohl mit FOR-NEXT oder der Kombination WHILE-WEND realisieren:

```
For i = 1 TO 10
 Print i
Next
```

Amiga-Basic ist natürlich auf den Amiga zugeschnitten und unterstützt ihn durch leistungsfähige Befehle:
- Grafik
- Menüs
- Maus
- Sound
- Animation
- Interruptkontrolle

Sogar spezielle Befehle, um auf Betriebssystem-Routinen zuzugreifen, stehen zur Verfügung.

Dennoch ist Basic nicht für alle Anwendungen die optimale Sprache. Es gibt keine Standards. Programme von anderen Computern können nicht ohne Änderungen verwendet werden. Selbst für den Amiga gibt es unterschiedliche Versionen:
- ABasic
- Amiga-Basic
- True-Basic
- SAM-Basic
- GFA-Basic (in Vorbereitung)
- A/C-Basic-Compiler für Amiga-Basic

(siehe auch Vergleichstest in dieser Ausgabe, Seite 114)

Systemnahe Programme lassen sich mit Basic nicht schreiben. Basic ist nicht sonderlich schnell. Es ist eine Interpretersprache. Mittlerweile existiert jedoch ein Basic-Compiler für den Amiga. Der A/C-Compiler übersetzt die in Amiga-Basic geschriebenen Programme. Doch die Geschwindigkeit kann sich noch nicht mit Assembler messen.

Assembler

Assembler ist eine maschinenorientierte Sprache. Der Quellcode eines Assembler-Programms hat große Ähnlichkeit mit dem Maschinencode. Assembler ist im Gegensatz

zu den Hochsprachen eine systemabhängige Programmiersprache. Für den Amiga gibt es verschiedene Assembler-Versionen:
— Makro-Assembler
— Seka-Assembler
— Profimat
— Devpac-Assembler
— Quello-Assembler (68020/68881)

Diese unterscheiden sich nicht in der verwendeten Sprache, die abhängig vom Prozessor ist. Der Unterschied liegt im wesentlichen in der Bedienung. (Siehe beispielsweise AMIGA 8/9, Seite 34 oder 68000er, Ausgabe 5/87). Zum Kennenlernen gibt es auch eine Assembler-Version auf Public Domain-Disketten (Fred Fish-Disk 81).

Im Herzen des Amiga befindet sich ein 68000-Prozessor. Er kennt 56 Grundbefehle und zwölf Adressierungsarten. Als Operanden können Bytes, Worte und Langworte verwendet werden. Ihnen stehen acht Datenregister (d0-d7) und Adreßregister zur Verfügung (a0-a7). Jedes besitzt mit 32 Bit doppelte Wortlänge. Das Adreßregister a7 wird in der Regel als Stapelzeiger eingesetzt. Es existiert zweifach. Immer nur eins ist aktiviert, je nachdem in welchem Zustand der Prozessor arbeitet. Er kennt den User- und den Supervisormodus. Nur im Supervisormodus sind einige privilegierte Befehle erlaubt. Zum Beispiel, wenn ein Wert ins Statusregister geschrieben werden soll. Es ist 16 Bit lang und enthält drei Interruptbits zur Decodierung von acht Interruptprioritäten, sowie zahlreiche Bedienerflags. Setzen Sie beispielsweise das Trace-Bit, arbeitet der Prozessor im Einzelschrittbetrieb.

Assemblerprogrammierung ist aufwendig und kompliziert. Sie können in Assembler zwar auf alle Systemfunktionen direkt zugreifen, sehr schnelle Programme erzeugen und beispielsweise Interrupts und Traps programmieren, aber Sie müssen selbst die einfachsten Dinge, wie die Ausgabe einer Zeichenkette auf dem Bildschirm, selbst programmieren. Die Programmierung einer Datenverarbeitung kann zu einer Lebensaufgabe werden.

Assemblerprogrammierung übt trotz der Schwierigkeiten auf viele Programmierer einen hohen Reiz aus. Für den Amiga gibt es jedoch Compiler in anderen Sprachen, die auch schnellen Maschinencode erzeugen.

Gerade aufgrund der Abhängigkeit vom Prozessor und der Komplexität von Assemblersprachen sind Werkzeuge nötig, mit denen maschinennah programmiert werden kann. Werkzeuge, die aber dennoch unabhängig vom System sind. Hochsprachen, die zudem strukturierte Programmierung unterstützen und höherer Datenelemente unterstützen. Eine solche Sprache ist C.

C wie Cäsar

C ist in den Bell Laboratories entwickelt worden, um ein Betriebssystem für 16-Bit-Rechner zu schreiben. Das Betriebssystem »Unix« wurde zu einem Standard. Auch das Betriebssystem des Amiga ist in C geschrieben worden. Dies unterstreicht die Leistungsfähigkeit der Sprache. Bemerkenswert ist der geringe Grundwortschatz von C, der nur 30 Schlüsselwörter umfaßt.

»if« und »while« dienen zur Steuerung. Die Anweisungen verhalten sich so, wie in den meisten bekannten Sprachen.

Mit »while« werden Schleifen programmiert. Vor jedem Durchlauf wird eine Bedingung getestet, beziehungsweise geprüft, ob ein Wert ungleich Null = »wahr« ist.

```
while <Ausdruck>
<Anweisung>
```

Soll eine Schleife mindestens einmal durchlaufen werden, muß die Bedingung erst am Ende abgefragt werden. Dies geschieht mit den Befehlen »do.....while«.
»switch« ähnelt dem Basic-Befehl ON GOTO.

```
switch(zahl)
{
   case 3:
     printf("Zahl=3" n);
     break;
   case 2:
     printf("Zahl=2" n);
   case 1:
     printf("Zahl=1" n);
     break;
   default:
     printf("andere Zahl
      " n);
}
```

Je nach Wert der Variablen i springt das Programm an die spezifizierte Stelle des Verteilers.

Eine typische »for«-Schleife sieht in C so aus:

```
for (i=1;i<n;i++)
 printf(%d n,i));
```

Die Laufvariable beginnt mit dem Wert 1 und erhöht sich bei jedem Durchgang um eins. i wird auf dem Bildschirm ausgegeben. Die Schleife wird abgebrochen, sobald i nicht mehr kleiner als n ist. Beachten Sie, wie einfach es ist, in C eine Variable zu inkrementieren.

All diese Befehle haben für jede C-Version Gültigkeit. Es gibt für C zwar noch keinen Standard, dennoch sind Programme leicht auf andere Computer übertragbar. Sie können beispielsweise ohne große Änderungen auch Listings der beiden C-Versionen für den Amiga von Lattice und Manx austauschen. Beide richten sich nach einem von Brian Kernighan und Dennis Ritchie geschriebenen Buch: »The C Programming Language«. Dieses auch ins Deutsche übersetzte Lehrbuch hat sich zum Quasi-Standard entwickelt.

Beide C-Compiler unterstützen folgende Datentypen:
— char Zeichen
— int Ganzzahl
— float Gleitkommazahl

Für Zahlen gibt es zusätzliche Typen:
— short [int]
— long [int]
— unsigned [int]
— unsigned short
— unsigned long
— double [Gleitkommazahlen mit doppelter Genauigkeit]

Neben den genannten gibt es auch zusammengesetzte Datentypen. Eine »Struktur« ergibt sich aus der Kombination mehrerer Variablen. Die Elemente einer Struktur müssen nicht vom gleichen Typ sein. Strukturen erleichtern die Bearbeitung von Objekten, die durch unterschiedliche Parameter beschrieben werden.

Sie müssen jede Variable vor Ihrer Verwendung einem der Datentypen zuordnen. Erst dann können Sie ihnen Werte zuweisen und damit rechnen.

C unterstützt fünf arithmetische Operatoren: Die vier Grundrechnungsarten und eine Rest-Funktion bei Ganzzahldivision. Nur fünf? Gibt es keine Exponentialfunktion?

Es gibt tatsächlich keine weiteren Rechenoperatoren in C. Aber alle benötigten Rechenarten sind in einer Standardbibliothek mathematischer Funktionen enthalten.

Lesehalle für C

Ein wesentliches Merkmal von C ist die Verwendung von Funktionen und Funktionsbibliotheken. Jedes Programm kann in kleine Module zerlegt werden, die eine eigenständige Funktion bilden. Das Hauptprogramm erhält den Namen »_main«, jede Funktion ihren eigenen Namen. In Bibliotheken stehen zahlreiche Standardfunktionen zur Verfügung. Ein Beispiel:

```
printf();
```

dient zur Bildschirmausgabe.

Jeder Funktion können Sie die Werte mehrerer Variablen zur Bearbeitung übergeben (Call by Value). Zum Beispiel ein Zeichen für »printf()«, das auf dem Bildschirm ausgegeben werden soll. Sie übergeben nur den Wert, nie die Variable. Wenn Sie eine Variable durch eine Funktion verändern möchten (Call by Reference), müssen Sie mit Zeigern arbeiten. Zeiger, die auf den Speicherplatz weisen, an dem sich eine bestimmte Variable im Speicher befindet. Diese Option macht unter anderem das Arbeiten mit C maschinennah. Zudem gibt es zahlreiche bitweise logische Operatoren. Ein bitweiser Rechts- oder Linksshift eines Byte, an einer durch einen Zeiger festgelegten Speicherstelle, ist genauso realisierbar, wie in Assembler.

In C stehen Ihnen nicht nur die Standardbibliotheken des C-Compilers zur Verfügung. Sie können auch alle »Libraries« des Amiga-Betriebssystems nutzen. Welche Routinen dort enthalten sind und wie sie arbeiten, ist ausführlich in den »ROM Kernal-Manuals« zum Amiga dokumentiert (Erschienen bei Addison-Wesley).

Die Geschwindigkeit zählt

Das Programmieren in C wird durch zahlreiche Hilfsmittel unterstützt:
— Es gibt spezielle Texteditoren, die besonders für die Erstellung von C-Quelldateien geeignet sind.
— »Key to C« besteht aus einer Sammlung von 125 C-Funktionen, vorwiegend zur Datenmanipulation.
— »LINT« überprüft Ihre Quelldatei vor der Übersetzung auf Fehler.
— »PowerWindows« nimmt Ihnen die Arbeit zur Programmierung von Menüs ab.

Gerade für C gibt es auch sehr viel Software auf Public Domain-Disketten.

Public Domain ist eine wertvolle Quelle für jeden C- und Assemblerprogrammierer. Beispielsweise finden Sie auf den

ersten »FishDisk« viele praktische Listings, die gut dokumentiert sind und den Einstieg in die C-Programmierung auf dem Amiga erleichtern.

Denn das Programmieren in C ist nicht so einfach, wie zum Beispiel in Basic. C ist eine sehr großzügige Sprache, die dem Programmierer viele Freiheiten gibt. Das bedingt aber auch, daß Sie sehr viele Fehler machen können, auf die der Compiler nicht aufmerksam macht. Es ist eine Sprache, die Programmiererfahrung erfordert und nicht unbedingt für Anfänger geeignet ist.

Pascal = Struktur

Pascal ist für Einsteiger leicht zu erlernen. Unter diesem Gesichtspunkt hat Niklaus Wirth die Sprache Anfang der 70er Jahre entwickelt.

Mit Pascal lernen Sie strukturiert Programmieren. Pascal ist wie C blockorientiert, das heißt Sie können ohne Sprungbefehle (GOTO) auskommen. Es ist ebenfalls erlaubt, Datenstrukturen zu benutzen und zu definieren. Die den Strukturen von C entsprechenden Datentypen heißen »Records«.

In Pascal sind Sie gezwungen Ihr Programm zunächst vollständig auf Papier zu entwerfen. Gute Programmierer erstellen ein Flußdiagramm, beziehungsweise ein Struktogramm, in dem der genaue Aufbau eines Programms grafisch festgelegt wird.

Variablen müssen direkt im Anschluß an den Programmkopf, der den Namen des Programms enthält, deklariert werden. Noch wichtiger, alle Unterprogramme — in Pascal als Prozeduren und Funktionen bezeichnet — werden vor dem eigentlichen Hauptprogramm definiert.

Dies ermöglicht vor allem schnelle und einfache Compiler, die den Quellcode in einem Durchlauf übersetzen. Auch der Pascal-Compiler von Metacomco für den Amiga ist ein »Einpass-Compiler«. Amiga Pascal richtet sich nach dem ISO-Standard (6. Entwurf 1981). Der Vorteil der Standardisierung ist: Alle Programme, die dieser Norm entsprechen, können auch auf dem Amiga eingesetzt werden.

BEGIN und END dienen dazu, den Anfang und das Ende eines Anweisungsblocks zu kennzeichnen, analog zu den geschweiften Klammern in C.

Ein komplettes Pascal-Listing, das die Zahlen von 1 bis 10 auf dem Bildschirm ausgibt:

```
PROGRAMM fordemo(output);
VAR
 i,j:INTEGER;
BEGIN
 j:=10;
 FOR i:=1 TO j DO
 BEGIN
 WRITE(i);
 END;
 WRITELN
END
```

Im Kopf des Programms steht der Name. Es folgt der Deklarationsteil. Auch jedes Programm besteht aus einem Block von Anweisungen und beginnt mit BEGIN. Der Zuweisungsoperator in Pascal sieht etwas ungewohnt aus »:=«. Eine FOR-Schleife enthält zusätzlich das Schlüsselwort DO. END steht am Schluß des Anweisungsblocks.

Selbstverständlich stehen Ihnen noch weitere Schleifenbefehle zur Verfügung.

```
WHILE <Ausdruck>
DO <Anweisung>
```

Der Befehl WHILE arbeitet wie in C. Mit REPEAT und UNTIL können Sie ebenfalls eine Schleife programmieren, die mindestens einmal durchlaufen wird. Die Verwendung von IF, THEN und ELSE ist praktisch dieselbe, wie in C oder Basic.

Eine Anweisung zum Programmieren eines Verteilers ist CASE:

```
CASE i OF 3: b:= i + 3;
          2: b:= i + 5;
          1: b:= i + 8;
END
```

Eine Schwäche: Es gibt keine Anweisung, die ausgeführt wird, wenn keiner der beschriebenen Fälle zutrifft.

Ähnlich wie in C wird der Befehlssatz von Pascal durch zahlreiche vordefinierte Standardprozeduren ergänzt.

Verwenden Sie beim Compilieren die Option »EXTEND«, eröffnen sich zusätzliche Funktionen: Beispielsweise können Sie die Befehle INCLUDE und EXTERNAL verwenden:

```
INCLUDE <File-Name>
```

erlaubt es dem Programmierer während des Compilierens zusätzliche Programmdateien in die Quell-Datei einzubinden. Noch bedeutender ist EXTERNAL. Dieser Befehl deklariert Unterprogramme als »extern«. Die Definition erfolgt außerhalb des Programms. Als externe Routinen können Sie Systemroutinen verwenden.

Die Objektcode-Dateien, die der Pascal-Compiler von Metacomco liefert, und die externen Module werden mit dem Linker »ALink« beziehungsweise »BLink« verknüpft. Durch diese Erweiterungen wird die Programmgestaltung flexibler als mit dem standardisierten ISO-Pascal.

Da das Standard-Pascal jedoch zum Beispiel keine Einbindung von maschinennahen Routinen erlaubte, und auch nicht das Compilieren getrennter Quelldateien, entwickelte Wirth ein neue modulare Sprache (MODUlar LAnguage).

Modula-2

Die endgültige Fassung der neu entwickelten Sprache lag 1980 vor: Modula-2.

Die Sprache wird schon heute als Nachfolger von Pascal bezeichnet, aller Schwächen bereingt: modular und maschinennah zugleich.

Sie können im Gegensatz zu Pascal mit Modula-2 einzelne Programmteile, Prozeduren und Funktionen getrennt compilieren. So lassen sich selbst große Projekte verwirklichen. Wer modular programmiert, kann im Laufe der Zeit eine große Bibliothek von vorcompilierten Routinen schreiben. Wie mit einem Baukasten, lassen sich aus den »Software-Chips« neue Programme entwerfen.

Einzelteile können von mehreren Programmierern getrennt bearbeitet werden. Alle Module werden separat übersetzt. Jeder Programmteil muß lediglich den anderen bekannt sein. Hierzu ist eine »Software-Schnittstelle« erforderlich; die Definition des Moduls. Hier steht, welche Variablen und Prozeduren aus anderen Teilen importiert werden.

Im- und Export

Ebenso findet sich in der Definition die Information, welche Werte als Resultat exportiert werden.

Der zweite Teil jedes Moduls ist die Implementation. Sie enthält die Befehle und erledigt die Arbeit.

Sicher erlaubt auch C die Modularisierung, jedoch nicht in dem Maße wie Modula-2. Weitere Unterschiede der Sprachen:

— Die Deklaration der Zeiger ist einfacher als in C.

— Außer Wert-Parametern kennt der Modula-2-Programmierer auch Referenz-Parameter.

— Zeiger sind in Modula an einen Typ gebunden.

(Einen Vergleich von Modula-2 und C finden Sie auf Seite 103 in dieser Ausgabe).

Modula-2 besitzt ebenfalls Vorteile gegenüber Pascal:

— Alle zusammengesetzten Anweisungen, die mit BEGIN anfangen, erhalten explizite Schlußsymbole. Das steigert die Lesbarkeit.

— Die Größe von Strings kann dynamisch geändert werden.

— Alle Funktionen und Prozeduren enden mit Return. So wird vermieden, daß eine Funktion uninitialisierte Resultate zurückliefert.

— Der Befehl CASE wurde mit der Option ELSE ausgestattet.

— Auf IF können beliebig viele ELSIF folgen.

— FOR kann mit beliebiger Weite programmiert werden.

Für den Amiga gibt es zwei Modula-Versionen, die alle auf ihn zugeschnitten sind:
TDI Modula II
M2 Modula II

Eine dritte Fassung erscheint in Kürze:
Benchmark Modula-2 by Oxxi, Inc.

In Modula-2 können Sie maschinennah programmieren. Es gibt unter anderem Befehle zum Lesen und Schreiben in die Hardware-Register. Wirth hat sogar einen Computer entwickelt, dessen gesamtes Betriebssystem in Modula-2 geschrieben ist. »Lilith« versteht nur Modula-2. Von Lilith stammt M2 Modula II ab. Einige Features diese Compilers:

— Innerhalb des Programms kann reiner Assemblercode eingebunden werden.

— ROM-Routinen lassen sich problemlos aufrufen.

— In Grundbibliotheken stehen zahlreiche fertige Module zur Auswahl.

Modula-2 hat eine große Zukunft, auch auf dem Amiga. Doch es gibt weitere Sprachen, die zunehmendes Interesse gewinnen. Programmiersprachen, die sich von den bisher beschriebenen unterscheiden.

Lisp (LISt Processing) und Prolog (PROgammierung in LOgik) sind deklarative, auch nicht nichtprozedurale genannte, Programmiersprachen. Was bedeutet das?

Diese Sprachen sind nicht anweisungsorientiert. Es gibt keine Funktionen oder Prozeduren. Bei prozeduralen Sprachen müssen Sie sich zur Lösung eines Problems einen Weg überlegen, einen Algorithmus. Diesen setzen Sie in ein Programm um. Der Computer berechnet dann auf dem beschriebenen Weg die Lösung. Anders in Prolog und Lisp. Hier beschreiben Sie das

Prolog und Lisp

Problem möglichst genau. Sie fertigen eine Liste von Ausgangsbedingungen, den »clauses«, an und wenn die Aufstellung komplett ist, geben Sie ein »goal« als Ziel an. Ein »Programm« besteht im wesentlichen aus einer Sammlung von Regeln, Fakten und Bedingungen. Der Computer ermittelt durch logische Verknüpfung aller Ausgangsdaten, ob es zu Ihrem Ziel eine Lösung gibt. Das Ganze geht weit über die Eigenschaften einer normalen Datenverarbeitung hinaus. Sie haben nicht nur die Möglichkeit, Daten abzufragen, die vorher eingegeben worden sind, sondern auch Schlußfolgerungen aus den Daten zu ziehen.

Sherlock Prolog

Vergleichen Sie den Computer mit einem Detektiv. Sie geben ihm alle Daten zu einem Kriminalfall, anschließend fragen Sie nach dem Ziel, dem Täter. Der Computer muß viel lernen. Wissen ist Macht. Wenn er genug weiß, kann er eine Lösung berechnen — abgesehen davon, daß der Täter immer der Gärtner ist —. Der gesamte Vorgang des Sammelns von Wissen, Erfahrungen und Zusammenhängen, verbunden mit der Fähigkeit logische Schlüsse aus den Daten zu ziehen, wird als Künstliche Intelligenz bezeichnet.

Prolog wurde bereits 1970 entwickelt. Zunächst nur an Hochschulen verbreitet, rückte die Sprache erst seit etwa 1980 in den Mittelpunkt des Interesses. Prolog ist die Sprache, die in Japan auf den Computern der fünften Generation eingesetzt wird. Einsatzgebiete sind zum Beispiel die Erkennung der menschlichen Sprache, die Analyse chemischer Strukturen, die Verifizierung naturwissenschaftlicher Gesetze, Strategiespiele und Expertensysteme. Expertensysteme sammeln zu einem Fachgebiet alle Erfahrungen, Messungen und Beziehungen — einfach alles. Das gesamte Wissen und alle daraus ableitbaren Ergebnisse können vom Bediener abgefragt werden.

Wie kommt die Maus aus dem Irrgarten?

Einen Algorithmus, um einen Weg durch einen Irrgarten zu finden, erfordert einen großen Aufwand. In Prolog läßt sich dies einfacher realisieren. Beschreiben Sie den Irrgarten, alle Wände, Türen, Wege und Verzweigungen. Geben Sie den Startpunkt ein und das Ziel. Mit Hilfe der implementierten Lösungsverfahren sucht Prolog den Weg. Eines dieser Verfahren ist das Backtracking.

Der Computer sucht auf einem bestimmten Weg einen Ausweg. Bei einer Verzweigung, einem Knoten, wählt er einen der Äste und verfolgt diesen weiter. Gelangt er an ein Hindernis, springt er an den letzten Knoten zurück und sucht auf dem zweiten Pfad. Das klingt nicht schwer, aber in C oder Basic erfordert eine Programmierung dieses Problems viel Arbeit. Prolog sucht sein Ziel automatisch.

Lisp ist eine ältere Sprache. Auch Lisp arbeitet mit Listen. Die kleinsten verwalteten Einheiten heißen Atome. Ein Atom kann nicht mehr geteilt werden. Es kann sich um eine Zahl oder einen Namen handeln. Eine Liste ist eine Struktur. Sie kann in ihre Elemente — Atome oder weitere teilbare Listen — zerlegt werden. Alle Atome dienen ebenfalls der Beschreibung von Objekten und deren Korrelationen. Anders als in Prolog sind aber nicht so viele automatische Lösungsverfahren eingebaut. Die neueren Lisp-Dialekte enthalten viele zusätzliche Befehle prozeduraler Sprachen, die aus Lisp eine sowohl anweisungs-, funktions- als auch elementorientierte Sprache machen. Recht komplex, vielfältig einsetzbar, aber nicht gerade etwas für Einsteiger.

Für den Amiga gibt es Cambridge Lisp von Metacomco. Eine Prolog-Version für den Amiga ist bereits angekündigt, aber noch nicht erhältlich.

Neben den neueren Sprachen, die in den letzten 15 Jahren entwickelt worden sind, um den gestiegenen technischen Anforderungen gerecht zu werden, existieren noch zahlreiche »ältere« Sprachen.

Fortran (FORmula PROzessing) ist die älteste Hochsprache überhaupt. Bereits 1957

wurde die Sprache speziell für mathematische Berechnungen programmiert. Im Ingenieur- und Hochschulbereich gibt Fortran auch heute noch den Ton an. Für jede Anwendung stehen große Programmpakete zur Verfügung. Gemessen an den modernen Sprachen hat Fortran jedoch einige Schwächen. Programme sind vollkommen unstrukturiert. Komplexe Datentypen können nicht bearbeitet werden. Eine Erweiterung stellt das 1977 vorgestellte Fortran 77 dar. Auch für den Amiga gibt es eine Version. A/C Fortran 77 erlaubt unter anderem den Zugriff auf die System-Bibliotheken des Amiga. Eine spezielle Version ist für einen Amiga mit 68020-Prozessor und 68881-Coprozessor geeignet.

Oldtimer

Neben Fortran spielt Cobol eine große Rolle. Es ist eine Sprache für die Programmierung kommerzieller Anwendungen, »die Sprache der Kaufleute«. Für andere Anwendungen ist es weniger geeignet.

Kraftvoll: FORTH

Forth ist eine Interpreter-Sprache. Es gibt auch Kombinationen aus Compiler und Interpreter. Multi-Forth heißt es für den Amiga. Forth verfügt unter anderem über gute Grafikbefehle. Ein wichtiger Punkt dieser Sprache: Alle Berechnungen erfolgen in der umgekehrten polnischen Notation.

Die Aufgabe 2*4+5 müssen Sie so eingeben:

```
2 4 5 * +
```

Auf vielen Heimcomputern macht Logo von sich reden. Vor allem durch die Implementation der Schildkröten-Grafik (Turtle). Logo zeichnet sich durch die einfache Erlernbarkeit aus und ist gut für Einsteiger geeignet. Schwerpunkte dieser Sprache sind Grafikprogrammierung und Audio-Anwendungen. Das sind Domänen des Amiga. Für den Amiga wäre eine Logo-Version daher wünschenswert, aber noch gibt es keine.

Eine interessante Mischung aus Basic, Logo und Pascal stellt Comal dar:
— Grafiken lassen sich leicht handhaben.
— Sie können strukturiert programmieren.
— Wie in Basic gibt es den Direktmodus.

Es soll aber in den Vereinigten Staaten an einer Fassung gearbeitet werden.

Guter Rat

Welche Sprache können wir Ihnen empfehlen. Einsteiger sollten auf jeden Fall mit dem guten Amiga-Basic beginnen. Mit Basic lernen Sie schnell. Es gibt ausreichend Literatur und viele Programme, die den Start in die Welt des Programmierens vereinfachen. Mit Amiga-Basic lassen sich nahezu alle Anwendungen realisieren. Erst wenn Sie an die Grenzen des Basic stoßen, sollten Sie einen Umstieg in Erwägung ziehen.

Wer Wert auf maschinennahe und strukturierte Programmierung legt, sollte sich mit C befassen. Auch hier steht ein großes Reservoir an Hilfsmitteln und fertigen Programmen für den Amiga zur Verfügung. Allerdings erfordert C gute Programmierkenntnisse.

Pascal ist als Lernsprache zwar eine Alternative zu Basic, doch Modula-2 bietet Ihnen mehr. Modula-2 ist auch eine Alternative zu C, hat aber noch nicht die Verbreitung auf dem Amiga.

Assemblerprogrammierung ist nicht für die Entwicklung von Anwenderprogrammen zu empfehlen. Tüftler, die den Computer »ausreizen« möchten, werden natürlich weiter auf Assembler schwören.

Wie wird sich die Entwicklung der Programmiersprachen fortsetzen? Im Zuge der technischen Revolution, der immer weiter steigenden Rechnerleistungen wird es auch neue Programmiersprachen geben. Die Computer werden immer komplizierter, schneller und verwalten mehr Speicher. Die Geschwindigkeit der einzelnen Sprachen tritt in den Hintergrund. Transputer verlangen neue Sprachkonzepte. Spracherkennung stellt hohe Anforderungen.

In einem oder zwei Jahrzehnten unterhalten wir uns mit dem Computer, benötigen weder Tastatur noch Maus, und Probleme werden im Dialog mit der Maschine gelöst. (ub)

Anbieter:
1. Alphatron, Luitpoldstr. 22, 8520 Erlangen, Tel. 09131/25018
2. ASH Computerware, Von-Brandl-Str. 15, 8229 Laufen/Salzach, Tel. 08682/1327
3. Atlantis, Ernst-Reuter-Str. 151, 5030 Hürth, Tel. 02233/31066
4. B Soltau Electronics KG, Esplanade 39, 2000 Hamburg 36, Tel. 040/340445
5. C.A.S., Sprendlinger Landstr. 71, 6050 Offenbach am Main, Tel. 069/842013
6. CompuStore, Fritz-Reuter-Str. 6, 6000 Frankfurt/Main 1, Tel. 069/567399
7. Data Becker, Merowingerstr. 30, 4000 Düsseldorf 1, Tel. 0211/3101030
8. DTM Werbung & EDV GmbH, Bornhofenweg 5, 6200 Wiesbaden, Tel. 06121/507999
9. Instant Replay, Pliszstr. 38a, 4130 Moers 3, Tel. 02841/73833
10. Interplan, Nymphenburger Str. 134, 8000 München 19, Tel. 089/1234066
11. ITC, Kornstr. 34, 4154 Tönisvorst 1, Tel. 02151/803031
12. Markt & Technik Verlag, 8013 Haar, Hans-Pinsel-Str. 2, Tel. 089/4613-0
13. PDC, Louisenstr. 115, 6380 Bad Homburg, Tel. 06172/24748
14. Firma J. Pfotenhauer, Postfach 1267, 7590 Achern, Tel. 07841/5056
15. Philgerma GmbH, Ungererstr.42, 8000 München 40, Tel. 089/395551
16. Softwareland, Franklinstr. 27, CH-8050 Zürich, Tel. 0041-13115959
17. Soyka Datentechnik, Hattingerstr. 685, 4630 Bochum 5, Tel. 0234/411913
18. SWS Computersysteme, Rachelweg 10, 2371 Hamdorf, Tel. 04332/1634
19. Video Loft Film, Fiedlerstr. 22-32, 3500 Kassel, Tel. 0561/877928
20. Jumbo-Soft, Horemansstr. 2, 8000 München 19, Tel. 089/1234065
21. A. + L. Meier-Vogt, Im Späten 23, CH-8906 Bonstetten

Programmiersprache	Hersteller	Preis (Mark)	Anbieter
Assembler:			
Makro-Assembler	Metacomco	169-398	1,3,4,8,9,10,11,14,15,16,18,19,20
K-Seka-Assembler	Kuma	137-248	1,2,17
Profimat	Data Becker	99	7
Devpac Assembler	Hisoft	148	12
Quello Assembler (68000)	Quello Inc.	249	11
Quello Assembler (68020)	Quello Inc.	369	11
Basic:			
TrueBasic	True Corp.	193-275	3,4,5,9,11,13,14,15,16,19,20
A/C Basic-Compiler	Absoft	375-798	3,6,13,15,16,17,20
C:			
Lattice C	Metacomco	324-450	1,2,3,4,8,9,10,11,15,16,17,18,19,20
Aztec C Professional	Manx	445-690	3,4,9,10,11,15,16,17,20
Aztec C Developer	Manx	578-798	s.o.
Aztec C Commercial	Manx	963-1320	s.o.
Forth:			
Multi Forth	Creative Solut.	598	3,9
J-Forth	Delta Research	298	3,16
Fortran:			
A/C-Fortran 77	Absoft	570-758	3,4,10,11,15,16,17
Lisp:			
Cambridge Lisp	Metacomco	390-650	2,3,4,8,9,10,11,15,16,19,20
Modula 2:			
TDI Modula II Standard	TDI	177-348	1,2,3,4,6,9,10,11,13,15,16,17,20
TDI Modula II Developer	TDI	297-505	s.o.
TDI Modula II Commecial	TDI	597-848	s.o.
M2 Modula II	Meyer-Vogt	21	
Pascal:			
Pascal	Metacomco	185-450	1,2,3,4,8,9,10,11,13,15,16,17,19,20
USCD-Pascal	Pecan/TDI	167-398	1,6,10,16,17,20
Hilfsprogramme:			
Lint	Gimple-Soft	284	6
Key to C	DRP	148	3
PowerWindows	Inovatronics	187-209	3,8,9,16

Diese Programmiersprachen sind für den Amiga bei diversen Anbietern erhältlich

Abenteuer leichtgemacht

Haben Sie schon einmal ein Adventure in die Ecke geworfen, weil Sie einfach nicht weiterkamen? Hier geben wir Ihnen einige grundsätzliche Tips, wie man Probleme, die sich dem Abenteurer stellen, angeht.

Ein Adventure ist eine besondere Spielform, die eine ständig wachsende Zahl von Anhängern findet. Für Einsteiger ist oftmals der Frust mit solchen Spielen ziemlich groß. Das liegt jedoch nur daran, daß es auch für Adventures bestimmte Regeln und damit verbundene Vorgehensweisen gibt, die man kennen und einhalten sollte. Wer sich allgemein schwer tut mit Abenteuerspielen, sollte sich an die nun folgenden Tips halten, mit denen auch Adventure-Profis an die Lösung kniffliger Aufgaben herangehen.

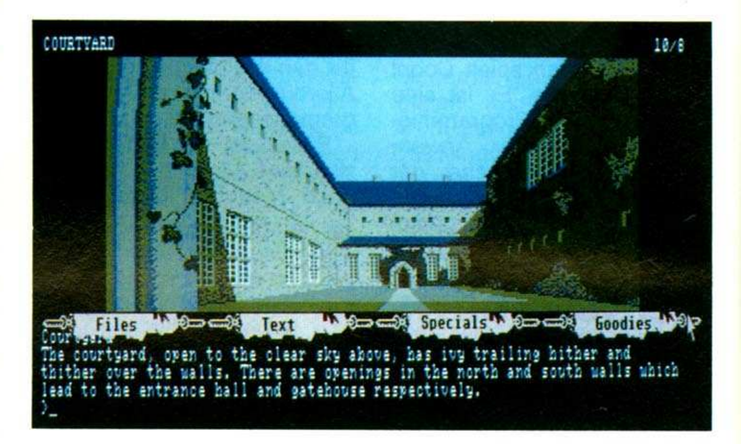
Da man die meisten Abenteuer durch die Eingabe von Texten (Befehle oder ganze Sätze) steuert, sollte bei fremdsprachigen Spielen ein Lexikon zur Grundausstattung gehören. Auch für Profis tauchen immer wieder spezielle Wörter auf, von denen man alle Bedeutungen kennen muß und nicht nur die aus der Schule bekannte. Besondere Wörter kann man sich je nach Bedarf aufschreiben und vielleicht in anderen Abenteuern wieder verwenden. Auch wenn viele Adventures einen großen Wortschatz haben, der den Spielfluß merklich verbessert, kann man als Einsteiger zuerst einmal mit einfachen Zwei-Wort-Kommandos auskommen. Jeder Parser (Programmteil, der die Texteingaben auswertet) wird auch simple Sätze verstehen. Sollte man später auf Probleme treffen, die sich nicht so einfach lösen lassen, muß nach und nach auf mehr Wörter und anspruchsvollere Satzkonstruktionen übergegangen werden. Den Wortschatz kann man, soweit er nicht extra in der Programm-Verpackung aufgeführt ist, im groben Rahmen auch durch Ausprobieren herausfinden. Entscheidend ist dabei, welche Kommentare das Programm auf die Eingabe von Worten, die es versteht oder nicht versteht, ausgibt. Einer der gängigsten Kommentare ist: »I don't know the word. . .«. In diesem Fall kann man davon ausgehen, daß das verwendete Wort unter keinen Umständen im Spiel zu gebrauchen ist. Währenddessen sind Bemerkungen wie »What do you mean by. . .« oder »You can't do that now« ein Hinweis darauf, daß man schon solche Aktionen ergreifen kann, dies jedoch im Moment keinen besonderen Sinn ergibt oder durch Umstände der Spielhandlung nicht möglich ist.

Auf das Wort kommt es an

Sehr oft kann man sich auch an bestimmten Worten, die in Kommentaren oder Beschreibungen ausgegeben werden, weiterhangeln, um so auf noch nicht erkannte Zusammenhänge zu schließen.

Vorsichtig müssen Einsteiger nur mit Abenteuern sein, die sehr stark mit anderen selbständigen Charakteren arbeiten (Kommunikations-Abenteuer). Diese natürlich auch vom Computer gesteuerten Figuren müssen angeredet und zu Handlungen aufgefordert werden. Grundsätzlich einfacher ist es, wenn solche Figuren nicht auftauchen.

Ein gutes Adventure muß nicht unbedingt über eine Grafik verfügen. Meist ist die Grafik nur eine stimmungsvolle Zugabe. Wichtige Objekte sind nicht darauf zu sehen, während unwichtige Sachen oft einen Großteil des Bildes einnehmen und noch nicht einmal in den Beschreibungen auftauchen. Entscheidend ist jedes Wort des beschreibenden Textes. Alle dort aufgeführten Gegenstände müssen genauestens untersucht und notiert werden. Damit ist auch schon ein anderes wichtiges Utensil des Abenteurers angeführt: die Karte. Alle Räume, die auf dem Weg zum Ziel betreten werden, sollten kartographiert werden. Objekte oder außergewöhnliche Ereignisse sind darauf zu verzeichnen. Ohne eine übersichtliche Karte wird man Hinweise, die sich aus der Anlage des Spielplans ergeben, nicht erkennen. Vermutungen über Geheimtüren oder versteckte Wege lassen sich dann leichter treffen. Sehr wichtig wird eine solche Karte, wenn man in einen Irrgarten gelaufen ist. Viele Labyrinthe funktionieren nach einem gewissen Prinzip. Um dieses herauszufinden, kann auch eine Karte hilfreich sein. Ein anderer Trick ist das Wegwerfen von Gegenständen, die man bei sich trägt. So markiert man die einzelnen meist völlig gleich gestalteten Räume und erkennt leichter, wenn man im Kreis gelaufen ist und schon bekannte Orte wieder erreicht. Weiterhin sollte man, falls eine Anzeige über den erreichten Punktestand vorhanden ist, alle Aktionen vermerken, die zu einer Verbesserung führen.



Wenn die Probleme vor Ihnen aufragen, wie die Mauern eines Schlosses, lesen Sie diesen Artikel

Mitnehmen oder wegwerfen?

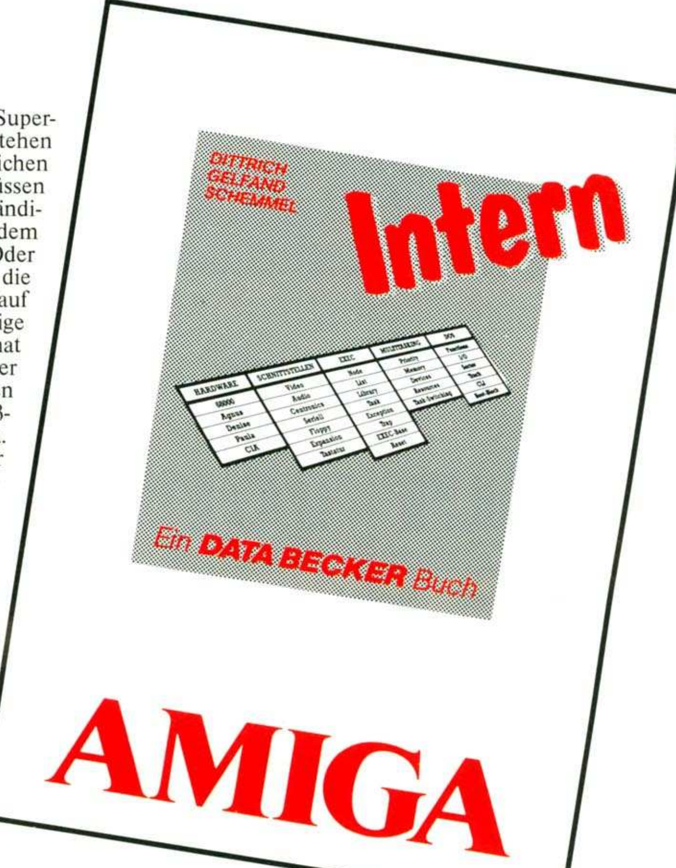
Einige Objekte, die sich im Lauf des Spieles finden, können eventuell Behälter sein und weitere Objekte oder Hinweise enthalten. Auch wenn sie nicht offensichtlich eine Kiste, Tasche oder ähnliches sind, so sollten sie doch daraufhin untersucht werden. Wenn das Programm nur eine bestimmte Anzahl an Gegenständen zuläßt, die die Spielfigur mit sich herumtragen kann, muß man sich wohl oder übel von einigen trennen. Dabei gilt die Regel, solche Gegenstände, die schon für die Lösung eines Problems gebraucht wurden, als erstes wegzuwerfen. Des weiteren kann man sich von reinen Hinweisen wie etwa Informationsblättern oder Briefen relativ problemlos trennen. Deren inhaltliche Aussage läßt sich auf einem Blatt notieren.

Besondere Sorgfalt muß man den gespeicherten Spielständen widmen. Vor einer wichtigen Entscheidung, insbesondere natürlich bevor man ein Labyrinth betritt, sollte das Spiel gespeichert werden. Falls das Programm keine besondere Kennzeichnung der Spielstände zuläßt, muß auch hier zur Selbsthilfe gegriffen werden. Eine kurze aber prägnante Bezeichnung für einen Spielstand muß man dann auf einem weiteren Notizzettel ausführlich mit Hinweisen auf die jeweilige Situation aufschreiben. Das erleichtert die Erinnerung an gespeicherte Spielsituationen, die schon wochenlang zurückliegen können. Manchmal geht die Lösung eines Adventures ja sogar über Monate hinweg nur Schritt für Schritt vorwärts. Wichtig ist, daß man dann den Überblick behält.

Als letzten Ausweg aus völlig festgefahrenen Abenteuern bietet sich der Kauf von Lösungstips des Programmherstellers an. Diese »Hint-Books« werden schon von vielen Firmen herausgegeben; meist liegt der Spielepackung ein Bestellschein bei.

Man sollte nur nicht der Verführung erliegen, alles auf einmal nachzuschauen. Das würde den Spielspaß erheblich mindern. (jk)

AMIGA INTERN.



Amiga-Anwender, die ihren Superrechner von Grund auf verstehen wollen, haben alle mit den gleichen Problemen zu kämpfen: Sie müssen sich durch mehrbündige, ausländische Bände arbeiten, die zudem nicht sonderlich aktuell sind. Oder aber sie besitzen den Ehrgeiz, die Geheimnisse ihres Rechners auf eigene Faust zu ergründen – einzige Voraussetzung hierbei: Man hat ein paar Jährchen Zeit. Einfacher ist es da schon mit dem neuen Amiga Intern: Hier finden Wißbegierige alles über den Amiga. In einem Band, in deutscher Sprache und absolut „up to date". Wie sehr Sie hier bis ins kleinste Detail vordringen, vermittelt bereits ein kurzer Blick ins Inhaltsverzeichnis: die Hardware des Amiga mit ausführlicher Beschreibung des 68000-Prozessors, der CIAS, der Customchips und der Schnittstellen, die Programmierung der Hardware in Assembler mit Speicherbelegung, Interrupts, Grafik und Sound. Doch Amiga Intern würde diesen Namen nicht verdienen, wenn es zusätzlich nicht noch jede Menge Fakten und Details zum Betriebssystem liefern würde. So erfahren Sie alles Wissenswerte zur EXEC: die wichtigsten Strukturen, Funktion und Arbeitsweise des Multitasking, I/O-Handhabung und Verwaltung der Ressourcen, EXEC-Base, resetfeste Programme und Strukturen. Fehlt nur noch das AmigaDOS. Auch hierzu nur einige Stichworte: Funktionen, Parameterübergabe, Fehlermeldungen, Boot-Vorgang, Aufbau der Diskette, IFF-Format, Programmstart von CLI und Workbench, interner Aufbau der CLI-Befehle, die Devices. Selbstverständlich finden Sie auch zahlreiche Beispiele zur Programmierung der EXEC- und DOS-Routinen. In Amiga Intern finden Sie den kompletten Rechner zwischen zwei Buchdeckeln.

Amiga Intern
Hardcover
639 Seiten, DM 69,–

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:

☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Name

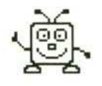
Straße

Ort

Für die Spieler nur vom Feinsten

Spiele wird es immer geben. Gerade der für den Heimbereich erschwingliche Amiga 500 läßt mit seinen tollen Grafik- und Soundfähigkeiten jedes Spielerherz höher schlagen. Wir stellen Ihnen die besten Spiele aus allen Bereichen vor.

Die Weihnachtszeit steht vor der Tür und damit auch wieder rosige Zeiten für den Verkauf von Spielen. Wer will sich nicht die langen Winterabende mit einem spannenden Spiel verkürzen? Wir haben auf den folgenden Seiten die schönsten und erfolgreichsten Spiele für den Amiga zusammengestellt. Sollten Sie eines der dort vorgestellten Spiele noch nicht kennen, liegt es nahe, sich dies zuerst einmal genauer anzusehen oder im Geschäft auszutesten, bevor Sie sich auf die Suche nach weiteren Programmen machen. Aus jeder Spielegattung haben wir zumindest eines besonders herausgestellt. Die Palette umfaßt Strategie-, Sport-, Denk-, Action- und Rollenspiele sowie natürlich auch Adventures und Simulationen. Für viele Sparten steht ein Spiel nur stellvertretend; es gibt noch andere ähnliche, die genausoviel Spielspaß verheißen. Darüber sollte man sich gegebenenfalls durch Nachfrage im Fachgeschäft oder durch Nachlesen in den Spieletests der Fachzeitschriften informieren. Die hier aufgeführten Spiele treten insbesondere durch ausgefallene Ideen hervor und motivieren somit gleichzeitig den Spieler. Außer bei reinen Textadventures, die völlig ohne Grafik und Sound auskommen, wird eine überdurchschnittliche Grafik und hervorragender Sound geboten.

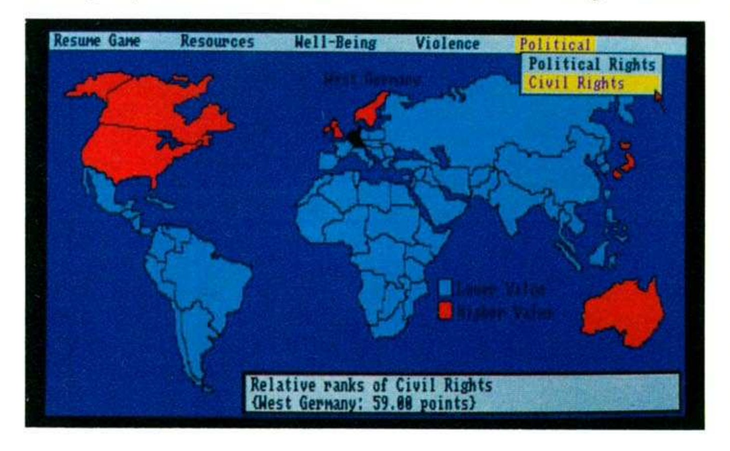


COMPUTERZEIT

Aber nicht das allein ist Bewertungskriterium für ein gutes Spiel. Der Bedienungskomfort muß sich auf einem hohen Niveau befinden. Für Actionspiele gilt dies in besonderem Maße für die Steuerung über den Joystick oder die Maus. Je weniger »kantig« die Abfrage vom Programmierer gestaltet wurde, um so mehr Spaß kommt später beim Spielen auf. Für ein Adventure ist dagegen der Parser, also der Programmteil, der die Texteingaben des Spielers auswertet, von entscheidender Bedeutung. Normalerweise hat der Amiga selbst mit nur 512 KByte Speicher genügend Platz, um einen guten Parser mit einem großen Wortschatz unterzubringen. Daß es trotzdem noch Adventures gibt, die kaum zu spielen sind, nur weil der Spieler ständig nach Worten suchen muß, spricht nicht gerade für die Programmierer. Dies kann man allerdings nicht auf den manchmal umfangreichen und auch schwierigen Wortschatz englischsprachiger Adventure schieben. Als Käufer sollte man sich darüber im klaren sein, daß man dann öfter im Lexikon nachschlagen muß. Kriterien sind hier der Umfang des Wortschatzes und die Schnelligkeit, mit der die Texteingaben des Spielers ausgewertet werden. Nicht zuletzt sollten natürlich auch entsprechend intelligente Reaktionen auf die Eingaben zu erwarten sein. Es gibt dabei die Möglichkeit, ein Spiel direkt im Geschäft vor dem Kauf anzutesten. Für Actionspiele bringt man am besten noch seinen eigenen Joystick mit. Oder man verläßt sich auf Erfahrungswerte mit ähnlichen Spielen aus dem gleichen Softwarehaus. Die letzte Möglichkeit sind dann noch die Spieletests aus der AMIGA oder beispielsweise Happy-Computer, die Aufschluß über die Qualität der Programme geben können. Dann stehen auf jeden Fall lange Abende voller ungetrübtem Spielespaß bevor. (jk)

Balance of Power

Was muß man tun, um die Welt vor einem Atomkrieg zu bewahren? Das ist die Ausgangsfrage, die sich in diesem Strategiespiel stellt. Als amerikanischer Präsident oder russischer Generalsekretär können Sie das mit Hilfe von »Balance of Power« einmal ausprobieren. Gleichzeitig müssen Sie aber das »geopolitische Image« des eigenen Landes vergrößern. Das ist gar nicht so einfach, da die andere Seite natürlich eine grundsätzlich verschiedene Einschätzung Ihrer Aktionen hat. Die wichtigste Arbeitsbasis ist dabei eine Weltkarte, wobei über jedes Land der Welt genaue statistische Daten und sogar Zeitungsmeldungen vorhanden sind. Von der Unterstützung der Guerillas eines Landes bis zum Einsatz von Soldaten ist fast alles möglich. Doch der Gegner könnte wegen einem Ihrer Schritte eine Krise ausbrechen lassen, was bis zum Atomkrieg führen kann. Genau diesen Atomkrieg soll man jedoch verhindern, denn wenn er ausbricht, gibt es keine tolle Grafik zu bewundern, sondern der Bildschirm wird abrupt schwarz. Der Programmierer dieser strategischen Simulation hat großen Wert darauf gelegt, alles sehr realistisch zu gestalten; der Schwierigkeitsgrad ist von »Anfänger« bis »Alptraum« variierbar. Die höchste Spielstufe stellt wahrlich einige Anforderungen an den Hobby-Strategen. Das Spiel wird normalerweise gegen den Computer gespielt, bietet aber auch zwei Spielern die Möglichkeit, miteinander um Einfluß auf der Welt zu verhandeln. Dabei können dann Verhandlungen, wie im richtigen politischen Alltag, zwischen den Spielern stattfinden. Um die umfangreiche Anleitung, die genau und ausführlich auf Probleme der aktuellen Weltpolitik eingeht, zu verstehen, sind aber gute Englischkenntnisse erforderlich.



The Bard's Tale

»The Bard's Tale« ist ein Fantasy-Rollenspiel, das bereits auf dem C 64 für Furore sorgte. Die Amiga-Version hat eine hervorragende Grafik und einige gute Soundeffekte bekommen. Es geht darum, sich zu Beginn eine Abenteuer-Gruppe aus verschiedenen Charakteren, die alle unterschiedliche Eigenschaften besitzen, zusammenzusuchen. Dabei sollten in dieser sechs Leute starken Gruppe sich ergänzende Charaktere vertreten sein, da jede besondere Fähigkeit im Spiel gebraucht wird. Einen Barden sollte man auf jeden Fall mitnehmen; er kann mit seiner magischen Harfe so manche Monster in die Flucht schlagen, und ohne ihn ist das Spiel nicht zu beenden. Zusammen ziehen die Abenteurer dann in die Stadt Skara Brae, um sie von dem Tyrannen Mangar zu befreien. Dazu müssen die Stadt sowie Labyrinthe, Tempel und Schlösser untersucht werden, um den Weg zu finden, der zu Mangar führt. Zuerst geht es nur ums blanke Überleben, doch nach einiger Zeit tritt die strategische Komponente mehr und mehr in den Vordergrund. Viele kleine Puzzles sind zu lösen, bevor der Barde zum Endkampf gegen den bösen Zauberer blasen kann. Da



das Spiel ziemlich kompliziert ist, braucht man einige Zeit, um alle Funktionen kennenzulernen. Die Amiga-Version bietet für Einsteiger die übliche Menübedienung und für Fortgeschrittene nochmals alle Funktionen über die Tastatur an. Wer Fantasy-Rollenspiele mag, wird von »The Bard's Tale« sicher lange gefesselt sein.



Chessmaster 2000

Eines der wenigen Schachprogramme, die momentan für den Amiga erhältlich sind, ist »Chessmaster 2000«. Es zeigt das Spielfeld entweder zwei- oder dreidimensional auf dem Bildschirm an, wobei durch die hervorragende Amiga-Grafik die Spielfiguren sehr realistisch dargestellt werden. Das Brett läßt sich auch rotieren und aus verschiedenen Richtungen betrachten. Bewegt werden die Figuren mit der Maus, wobei nicht mögliche Züge vom Computer automatisch abgelehnt werden. Gerade Schach-Anfänger werden es sehr zu schätzen wissen, daß der »Chessmaster« auf Wunsch alle Zugmöglichkeiten einer Figur anzeigt oder einen Zug vorschlägt. Fortgeschrittene Schachspieler können sich mit Hilfe des Programms bestimmte Aufgaben auf dem Brett vorgeben. Die Spielstärke des Programms läßt sich in zwölf Stufen einstellen, wobei man allerdings bei einer höheren Spielstärke eine lange Rechenzeit in Kauf nehmen muß. In der höchsten Spielstufe überlegt der Computer immerhin im Durchschnitt 6000 Minuten pro Zug. Die Überlegungsphase des Computers läßt sich aber auch beliebig abkürzen oder unterbrechen. Drei verschiedene Spielarten, nach denen der Computer beim Ziehen vorgeht, sind integriert. Der Stil geht von Normal über Schnellzüge bis zur Ermittlung des besten Zuges. Viele Besonderheiten wie ein Lernmodus, eine Eröffnungsbibliothek und die Anzeige der schon geschlagenen Figuren machen das Spiel sehr komfortabel. Als Gag kann ein Screen, der eine Tabellenkalkulation darstellt, vor die normale Anzeige des Spiels geschaltet werden, falls man in der Firma während der Arbeitszeit spielt und zufällig der Chef vorbeischaut.

Emerald Mine

Eigentlich wollten Sie ja nur Smaragde suchen gehen — aber jetzt müssen Sie sich in der »Emerald Mine« vor allen möglichen Dingen in Sicherheit bringen, die Ihnen das Leben dort schwermachen. Vom Spielprinzip her ist Emerald Mine eine wesentlich verbesserte und mit neuen Eigenschaften vollgestopfte Version des Spieleklassikers »Boulder Dash«. Mit viel Fingerspitzengefühl steuern Sie ein kleines Männchen durch die Mine, wobei zum Einsammeln der Smaragde nur wenig Zeit zur Verfügung steht. Erst wenn alle Smaragde gefunden wurden, öffnet sich der Ausgang in den nächsten Level. In den 80 verschiedenen Leveln dieses Actionspiels begegnet man aber einer Vielzahl von Monstern, Steinen, Sandsorten und sonstigen Kreaturen, die vom Spieler jeweils verschieden behandelt werden müssen. Doch nicht alle »Einwohner« der Mine sind Ihnen feindlich gesinnt — einer verwandelt zum Beispiel wertlose Steine in Smaragde! Obwohl »Emerald Mine« erstaunlich preiswert ist, bietet es durch die vielen Level und die witzigen Sounds eine Menge Spielspaß. Besonders interessant ist es, das Spiel mit zwei Joysticks und Spielern anzugehen, die sich dann beide gleichzeitig auf dem Bildschirm an schwierigen Aufgaben versuchen können. Manche Probleme können besser mit zwei Diamantensuchern, die zusammenarbeiten, gelöst werden. Gut gelöst ist außerdem, daß man das Spiel nach einem Neustart nicht ständig wieder von vorne beginnen muß. Unter dem zu Beginn eingegebenen Spielernamen wird nicht nur die erreichte Punktezahl auf Diskette gespeichert, sondern auch der Level, in dem man sich zuletzt befand und an dem man gescheitert ist. Sofort kann man wieder in noch interessanten Leveln weiterspielen.



Faery Tale

Es war einmal in einem fernen Land, da lebten drei junge Männer. Sie zogen aus, um den Talisman zu finden, der ihr Dorf bisher vor allem Bösen geschützt hatte und nun selbst von bösen Mächten geraubt worden war. Im »Faery Tale Adventure« (Märchen-Adventure) wird der Spieler in die Zeit der Märchen und Sagen zurückversetzt. Einer der drei Brüder wird vom Spieler gesteuert. Er zieht im Land herum und muß sich dabei mit allen möglichen zwielichtigen Gestalten herumschlagen. Hilfe bekommt er von den Zauberern, die im Land herumziehen. Wie bei jedem Rollenspiel muß auch hier erst eine gewisse Einstiegshürde überwunden werden. Wenn die Figur nämlich etwas stärker und besser ausgerüstet ist, kann man an die Lösung der eigentlichen Aufgabe gehen. Stirbt die Spielfigur, kann nur noch auf die Hilfe einer Fee gehofft werden. Hat man noch genügend Glück, was am unteren Bildschirmrand in Punkten angezeigt wird, so wird der Spieler wiederbelebt. Falls nicht, hat man noch zwei weitere Chancen. Damit nicht wieder ganz vom Nullpunkt gestartet werden muß, sollte man zuerst den Ort, an dem der vorher verstorbene Bruder liegt, aufsuchen. Dann können alle



Sachen, die dieser eventuell bei sich hatte, weiterverwendet werden. Das nicht allzu schwierige Spiel verlangt viel Geduld, kann bei Kämpfen aber auch sehr aktionsreich sein. Dazu trägt auch die sehr gut gemachte Musik bei. Durch die riesige Landschaft, die man erforschen kann, bleibt es für eine lange Zeit interessant.



Flight II

Für die meisten Heimcomputer gibt es eine Umsetzung des Flugsimulators »Flight Simulator II«. In der Amiga-Version zeigt sich die gewonnene Erfahrung der Programmierer von Sublogic. Sie können sich entweder eine Cessna oder einen Learjet ins heimische Wohnzimmer holen. Die Bedienung findet hauptsächlich über die Tastatur statt. Geflogen wird über fünf verschiedenen amerikanischen Gebieten, wobei die beiliegenden Navigationskarten beim Auffinden der Flughäfen helfen. Man fühlt sich wirklich wie in einem richtigen Flugzeug — neben den üblichen Instrumenten fehlen auch Navigationsinstrumente für den Blindflug nicht. Wer will, kann seinen eigenen Flug von einem Verfolgerflugzeug aus betrachten. Überhaupt wurde sehr darauf geachtet, die Simulation so realistisch wie möglich zu machen. Sogar die Wahrscheinlichkeit eines Motorausfalls ist einstellbar. Wer eher kriegerisch veranlagt ist, kann einen Luftkampf im ersten Weltkrieg simulieren lassen. Als besonderes Bonbon darf Flight II auch mittels Datenfernübertragung über ein Modem oder einen Akustikkoppler mit einem anderen Spieler, der weit entfernt ist, gespielt werden. Während des Spielens kann man dem Partner dabei sogar Textmeldungen zusenden. »Flight II« ist sicher eines der besten Simulationsspiele überhaupt. Auch Anfänger dürften beim Fliegen keine Probleme haben, wenn sie sich mit der Tastaturbelegung vertraut gemacht haben. Eine Landung erfordert allerdings schon einige Übung. Das sehr ausführlich gehaltene Handbuch vermittelt auch einiges über die Grundzüge der Sportfliegerei. Inzwischen gibt es sogar schon Zusatzdisketten mit weiteren Landschaften.

Ihr PC hat mehr zu bieten als Zahlen und Fakten:

FASZINATION



Entdecken Sie mit PC Magazin PLUS völlig neue Perspektiven einer bislang ausschließlich business-orientierten Computerwelt – der Welt der IBM-PCs und Kompatiblen.
Ihr Personal Computer kann viel mehr: Erleben Sie die wunderbare Welt der Grafik, das Abenteuer spannender Spiele, die Faszination des Programmierens. Machen Sie Ihren PC außerdem zum unschlagbaren Instrument Ihrer beruflichen Zukunft: Know-how, Wissenserweiterung und engagierte Weiterbildung.
Unser Kennenlern-Angebot bietet Ihnen ein kostenloses Probeexemplar.

Kennenlern-Angebot

mit einem kostenlosen Probeexemplar PC-Magazin-Plus

Ja, ich interessiere mich für PC-Magazin-Plus und möchte ein kostenloses Probeexemplar dieser Zeitschrift. Wenn ich PC-Magazin-Plus weiterlesen will, brauche ich nichts zu tun, ich bekomme dann PC-Magazin-Plus regelmäßig per Post zum günstigen Jahrespreis von 84,— DM (für 12 Ausgaben, Auslandspreise und Studentenabo siehe Impressum).

Geld-zurück-Garantie:

Ich kann das Abonnement jederzeit kündigen, es gibt keine Kündigungsfrist. Zuviel bezahlte Beträge erhalte ich zurück.

Name ______________________ Vorname ______________________

Straße ______________________ PLZ/Wohnort ______________________

Telefon ______________________ Datum, 1. Unterschrift ______________________

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestelladresse Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Str. 2, 8013 Haar bei München widerrufen kann. Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs an obige Adresse. Ich bestätige dies durch meine 2. Unterschrift.

Datum, 2. Unterschrift ______________________

A188

Garrison

In diesem Action-Spiel geht es um die Rettung einer schönen Prinzessin namens Angelique. Das Wundermittel zu ihrer Rettung und Heilung einer schweren Krankheit ist in der Burg eines finsteren Magiers zu finden. Also brechen fünf Helden auf, sich durch 128 verschiedene Etagen voller Monster zu kämpfen. Das Überleben in dem Schloß ist ganz schön schwierig, besonders wenn man dabei allen möglichen unliebsamen Gestalten begegnet. »Garrison« bietet deshalb etwas Einmaliges: Spielt man es zu zweit, können sich die beiden gerade spielenden Figuren gegenseitig helfen. Sie können aus den fünf verschiedenartigen Spielfiguren ausgesucht werden, die alle ihre Stärken und Schwächen haben. Das ist auch dringend nötig, denn einige Gegner lassen sich mit roher Gewalt bekämpfen, andere aber nur mit einem Zauberer. Besonders wichtig sind für die Spieler auch die verschiedenen im Schloß zu findenden Zaubersprüche und Schlüssel. Die unterschiedlichen Eigenschaften der Kämpfer kann man durch das Aufsammeln von Flaschen voll magischem Elixier verbessern. Man sollte einen Trank zur Steigerung der Schußkraft zum Beispiel seinem Elf zukommen lassen, da dieser zwar sehr schnell zu Fuß, aber auch sehr schwach ist. Während Speed-Potions eher für den schweren und langsamen Krieger bestimmt sind. Die Übermacht der Gegner macht dieses Actionspiel an vielen Stellen recht schwierig. Wirkliche Action-Profis werden aber viel Spaß mit Garrison haben. Die 128 verschiedenen Level, die nach dem Programmstart in einer zufälligen Reihenfolge geladen werden, lassen keine Langeweile aufkommen. Eine extra High-Score-Liste auf Diskette für jede einzelne Spielfigur ergänzt den guten Eindruck.



Guild of Thieves

Um in die Gilde der Diebe in Kerovnia aufgenommen zu werden, müssen Sie Ihre Fähigkeiten als Meisterdieb unter Beweis stellen. Sie brechen auf zu einer Insel, die an den verschiedensten Orten Schätze beherbergt. Diese Schätze sind jedoch nicht immer auf den ersten Blick als solche erkenntlich. Sie müssen alle wertvollen Gegenstände auffinden und entführen, ohne daß ihre Besitzer etwas davon merken. Doch das ist gar nicht so einfach, denn in ein Schloß muß man erst einmal hineinkommen, und nicht alle Wesen auf der Insel sind Ihnen freundlich gesinnt. Die einzelnen Schätze kann man natürlich nicht ständig mit sich herumtragen und muß sie deswegen in einem Banksafe deponieren. Letzte Aufgabe, bevor Sie dem Meisterdieb die Erfüllung Ihrer Aufgabe bestätigen können, ist es dann eben, diese Bank auszuräumen. »The Guild of Thieves« ist ein Grafikadventure der Spitzenklasse und zeichnet sich besonders durch die hervorragenden Grafiken aus, die zusätzlich zu den umfangreichen Beschreibungen der einzelnen Objekte angezeigt werden. Für unerfahrene Abenteuerspieler ist die Schwierigkeit der gestellten Aufgaben zum Teil enorm hoch. Ist man jedoch geübt im »Um die Ecke denken« und Suchen nach ungewöhnlichen Lösungen, wird man mit der Diebesgilde seine helle Freude haben. Sehr gute Kenntnisse der englischen Sprache braucht man nicht mitzubringen. Und im Falle eines Problems, bei dem man nicht weiterkommt, helfen codierte Tips in der Anleitung weiter.



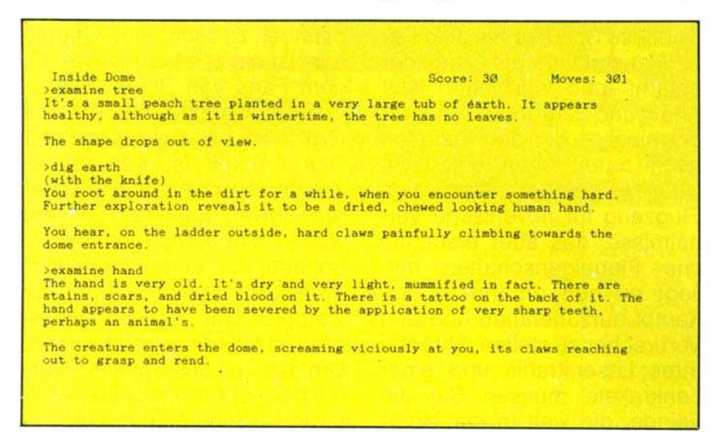
Leader Board

Falls Sie nicht zu den wenigen Leuten gehören, die sich »echtes« Golfspielen leisten können, bekommen Sie mit »Leader Board« einen Ersatz auf dem Computer. Bis zu vier Spieler können sich an einer Partie auf vier verschiedenen Golfplätzen beteiligen. Auch die Anzahl der »Holes« ist in vier Sufen wählbar von 18 über 36 und 54 bis hin zu 72 Löchern. Die reizvolle 3D-Grafik zeigt Ihnen dabei genau, wo Sie den Golfball mit möglichst wenig Schlägen hinschießen müssen. Nach Abschluß eines Kurses wird eine Übersicht der Spielerrangfolge angezeigt, das sogenannte Leader Board. Es führt natürlich derjenige, der am wenigsten Schläge für die gestellte Aufgabe benötigt hat. Dazu stehen Ihnen immerhin 13 verschiedene Schläger von Holz bis Eisen zur Verfügung, wobei die Auswahl manchmal ein wenig schwerfällt. Um nun den relativ komplizierten Schlagvorgang durchzuführen, muß die Maus gedrückt, losgelassen und wieder gedrückt werden. Damit beinflußt man die Schlagstärke sowie Schlaggeschwindigkeit und Schlägerstellung. Das richtige Timing erfordert einige Erfahrung. Mit ein bißchen Übung kommt jedoch schnell Stimmung auf. Anhand der drei Spielstufen steigert sich die Schwierigkeit, den Ball dorthin zu bringen, wo man es beabsichtigt hat. Einflüsse wie Wind und schräge Flächen machen den Spielern das Leben nicht leicht, tragen jedoch erheblich zur Realitätsnähe dieses von den Herstellern als Golfsimulation bezeichneten Spiels bei.

Wie alle Sportspiele macht »Leader Board« mit mehreren Mitspielern am meisten Spaß. Durch die verschiedenen Schwierigkeitsgrade bleibt das Spiel auch für Golfer mit mehr Erfahrung lange interessant.

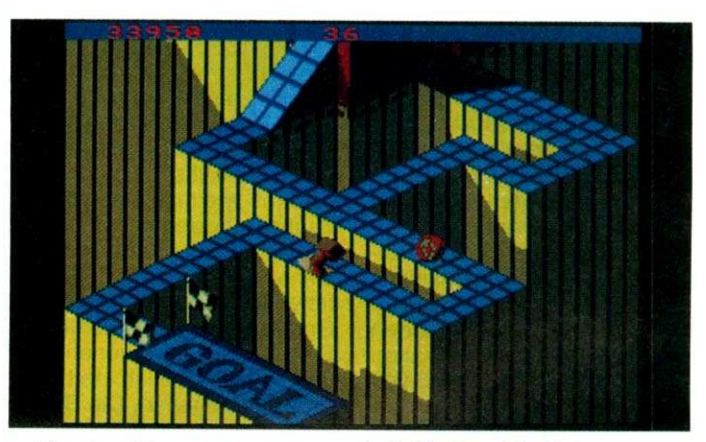
The Lurking Horror

Textadventure sind eine Spezialität von Infocom. Diese Abenteuer ohne jegliche Grafik und Sound beziehen ihre Faszination ausschließlich aus den stimmungsvollen Beschreibungen der Szenen und Situationen, dem großen Wortschatz und dem guten, intelligent gemachten Parser. Das Programm versteht über 1000 Worte und auch schwierige Sätze. In »The Lurking Horror« schlüpfen Sie in die Rolle eines Studenten, der sich mal wieder mit den Schulaufgaben verspätet hat. Also, rüber in die Uni, um die fällige Diplomarbeit dort noch über Nacht zu schreiben. Doch irgend jemand hat Ihre angefangene Arbeit gelöscht. Zudem tobt der Schneesturm draußen immer heftiger; an eine Rückkehr nach Hause ist vor dem nächsten Tag nicht zu denken. So bleiben Sie eben in der Uni. Doch sind die alten Gänge und Keller wirklich so ungefährlich wie es tagsüber scheint? Als Sie am Computer in einen unruhigen Schlaf mit Alptraum fallen, fangen die Horror-Ereignisse der Nacht erst richtig an. »The Lurking Horror« (das schleichende Grauen) ist ein Textadventure, das seinem Namen alle Ehre macht — die Uni wimmelt nur so von Monstern und sonstigen unfreundlichen Wesen. Die Schwierigkeit ist nicht zu hoch, so daß auch Anfänger, die einmal in ein Textadventure hineinriechen möchten, mit diesem Abenteuer ihre Freude haben werden. Wer einigermaßen gute Englischkenntnisse und Spaß am Lösen von logischen Rätseln hat, wird am Computer einige sehr gruselige Stunden verbringen. Obligatorisch für Infocom-Adventures ist die gut gemachte Verpackung, die wieder einige Tips enthält. Außerdem ist als Gimmick ein echtes schleimiges Monster beigelegt, das während des Spielens an den Bildschirm geheftet werden kann.



Marble Madness

Das Spiel mit den verrückten Murmeln war das erste Action-Spiel, das die Grafikfähigkeiten des Amiga richtig ausnutzte. Außerdem ist es die erste 1:1-Umsetzung eines Profi-Automaten aus der Spielhalle. Es liegt eigentlich eine ganz einfache Handlung zugrunde: Der Spieler muß eine kleine Kugel über ein Spielfeld in ein bestimmtes Ziel steuern. Zur Steuerung kann dabei sowohl der Joystick als auch die Maus herhalten. Doch es stellen sich ihm eine Fülle von Hindernissen in den Weg. Durch vorsichtiges Umfahren müssen etwa Kugelfresser daran gehindert werden, die kleine Kugel zu verspeisen. Eine Eisfläche am Ende des zweiten Levels behindert die Steuerung und kann nur mit viel Geschick überwunden werden. Paßt der Spieler nicht auf, gerät seine Murmel in einen aggressiven Säurefleck oder wird von einem Staubsauger in Stücke zerlegt. Da in jeder Spielstufe neue Hindernisse auftauchen, ist für Abwechslung gesorgt. Da sind zum Beispiel unangenehme Hämmer, die die eigene Murmel nicht so einfach des Weges ziehen lassen, oder gemeine schwarze Kugeln, die einen mit Vorliebe über die Kanten in die Tiefe schubsen. Wer will, kann auch zu zweit gleichzeitig das Rennen antreten. Der andere Spieler kann dann natürlich auch zur Gefahr werden und einen aus dem Spiel werfen. Die hervorragende 3D-Grafik trägt viel zum Spielspaß bei. Ein Geschicklichkeitsspiel also, das es in sich hat — auch Anfänger werden durch den nicht zu hohen Schwierigkeitsgrad viel Spaß mit Marble Madness haben.



Shanghai

»Shanghai« ist die moderne Version des alten asiatischen Brettspiels Ma-Jongh. Am Anfang des Spieles hat der Spieler dabei einen regelmäßig angeordneten Stapel mit Spielsteinen vor sich, den sogenannten »Drachen«, der aus 144 einzelnen Steinen besteht. Es gibt insgesamt 42 verschiedene Sorten von Spielsteinen. Der Spieler muß nun zwei gleiche Steine finden, die »frei« sind. Frei bedeutet in diesem Zusammenhang, daß der Stein von oben frei sichtbar ist und sich nach links oder rechts wegschieben läßt. Hat man ein Pärchen gleicher Steine gefunden, können beide weggenommen werden. Dies geschieht einfach dadurch, daß man erst einen Stein mit der Maus anklickt. Er erscheint dann angeleuchtet auf dem Bildschirm. Danach braucht der zweite passende Stein nur mit einem Doppelklick bedacht werden und schon sind beide Steine vom Bildschirm verschwunden. Ziel des Spieles ist es, am Ende möglichst wenig Spielsteine übrig zu haben. Wer es sogar schafft die Pyramide so aufzulösen, daß am Ende gar keine Steine mehr übrig sind, für den gibt es einen grafischen Leckerbissen. Ein Drachen durchbricht die leere Spielfläche und schaut einen mit bösen Augen an. Dieses Denkspiel kann allein oder mit mehreren Spielern gespielt werden. Besonderen Spaß bereitet es, mit zwei Mäusen gegeneinander unter Zeitlimit zu spielen. Die Regeln sind leicht zu begreifen. Um aber alle Steine zu entfernen, brauchen auch schnelle Denker viel Erfahrung. Durch den zufälligen Aufbau der Pyramide wird das Spiel auch nie langweilig. Übrigens läßt sich jede Pyramide auflösen. Der Computer bedenkt, daß sich nicht alle Steine einer Sorte übereinander befinden dürfen.

Starglider

Etwas Schreckliches ist passiert: Die Egronianer haben den Planeten der friedensliebenden Novenianer in Schutt und Asche gelegt und besetzt. Die Sentinels, die das Abwehrsystem bilden, wurden überlistet. Eigentlich lassen sie keine Feinde an die Planeten heran. Doch Herman Kruud, der Erzbösewicht, hat eine Schwachstelle erkannt und ausgenützt. Einzig Raumschiffe, die aussehen wie Starglider (eine Rasse von seltsamen Weltraumvögeln) werden von den Sentinels durchgelassen. Die Starglider unterliegen einem Ökologie-Schutzgesetz. Der fiese Schurke hat also seine Raumschiffe als Starglider getarnt und verbreitet jetzt Angst und Schrecken. Sie sind der einzige, der den Planeten noch retten kann. Mit dem »AGAV«, einem Museums-Flugzeug für novenische Verhältnisse, das aber erstaunliche Flugeigenschaften hat, liegt es nun an Ihnen, den Kampf aufzunehmen und den Verbrecher zu stellen. Mit Hilfe Ihres Laserstrahls und einer Lenkrakete müssen Sie die Feinde, die weit in der Überzahl sind, bekämpfen. Die einzige Hilfe von außen bekommen Sie in vier Einsatzbasen, in denen der Laser nachgeladen und das Raketendepot nachgefüllt wird. Ihr Ziel ist es, das Kommandoschiff der Egronianer, den Starglider, in dem sich der Kopf der Bösewichte befindet, zu zerstören. »Starglider« ist ein 3D-Actionspiel, mit dem Fans von Schießspielen viel Spaß haben werden. Die Grafik ist sehr gut gemacht: Das Cockpit ist normal gezeichnet, während die sichtbaren Bewegungen all dessen, was sich außerhalb des AGAV befindet, in einer 3D-Vektorgrafik blitzschnell dargestellt werden. Der Schwierigkeitsgrad ist allerdings relativ hoch. Um das Ziel des Spieles zu erreichen, brauchen selbst »Ballerprofis« eine Menge Zeit.



Uninvited

Eines Nachts haben Sie auf der Landstraße einen Unfall gebaut, weit abseits einer Ortschaft. Um Sie herum tobt ein Gewitter, außerdem ist Ihr Bruder, der mit im Auto war, verschwunden. Die einzige Rettung scheint ein unheimliches altes Haus zu sein, das sich in der Nähe der explodierten Trümmer Ihres Autos befindet. Nachdem Sie sich Einlaß verschaft haben, passiert das, was sonst nur in billigen Gruselfilmen vorkommt: Die Tür fällt hinter Ihnen ins Schloß und läßt sich nicht wieder öffnen. So müssen Sie das ganze unheimliche Haus durchsuchen, um den Ausgang zu finden und außerdem auch noch Ihren Bruder zu retten. Viele kleine, aber doch logische Rätsel sind zu lösen, bevor man erkennt, wie man einen bösen Zauberer besiegen und den eigenen Bruder aus den Klauen seiner dämonischen Helfer befreien kann. »Uninvited« ist ein Grafikadventure im wörtlichen Sinn, denn alle Aktionen werden ausschließlich mit der Maus eingegeben. Die Bedienung gestaltet sich wie das Arbeiten auf der Workbench. Gegenstände, die sich mitnehmen lassen, werden zum Beispiel einfach mit der Maus aus dem Szenenbild in das Fenster für Sachen, die man bei sich trägt, gelegt. Wer Grundkenntnisse der englischen Sprache hat und gerne nicht allzu schwierige Adventure-Rätsel löst, kommt bei »Uninvited« voll auf seine Kosten. Die witzigen englischen Kommentare und die gut digitalisierten Geräusche (wie etwa das Donnern des Gewitters oder das Knarren der Türen) machen das Spiel noch interessanter.





Winter Games

Selbst wenn Sie im richtigen Leben keine Weltrekorde im Wintersport aufstellen — »Winter Games« läßt Sie wenigstens auf dem Amiga zum Sieger werden. Komplett mit sehenswerter Eröffnungszeremonie und Trainings-Modus ist Winter Games immer noch eines der interessantesten Sportspiele für den Amiga. In sieben verschiedenen Disziplinen (Skisprung, Kunstspringen, Eisschnellauf, Bobfahren, Biathlon und zwei Arten von Eiskunstlauf) können Sie Ihre wintersportlichen Fähigkeiten unter Beweis stellen. Jede Disziplin erfordert dabei eine andere Taktik, um an die Punkte der Schiedsrichter heranzukommen. Dazu sind in der Anleitung die verschiedenen Techniken genau beschrieben. Alle sieben Sportarten sind grafisch sehr gut und mit Liebe zum Detail programmiert. Da kann es beim Skisprung schon einmal passieren, daß der Springer sich auf die Nase legt oder beim Kunstspringen kopfüber im Schnee steckenbleibt. Grafischer Höhepunkt ist jedoch eindeutig die Winterlandschaft, die beim Biathlon sauber im Hintergrund gescrollt wird. Fast alle Disziplinen haben eine (manchmal etwas eigenwillige) Musikbegleitung und diverse Sounds, die das Ganze merklich auflockern. Obwohl man für einen Weltrekord viel Übung braucht, sind die Grundzüge der einzelnen Disziplinen leicht zu erlernen. Damit ist »Winter Games« ein Sportspiel für die ganze Familie. Den meisten Spaß macht es sowieso, wenn mehrere Leute zusammen spielen. Ähnlich einer Olympiade kann sich dann jeder Spieler eine eigene Nationalität aussuchen und es gibt Siegerehrungen. Eine Liste der Weltrekorde, die auf Diskette gespeichert wird, läßt die richtige Wettkampfstimmung aufkommen.

Wir machen Preise!

Sie zahlen: keine Einfuhrumsatzsteuer, keine Verpackung, keine Versandspesen

HITS DES MONATS

DIGI VIEW	289,–
DIGI PAINT	98,–
VIDEOSCAPE 3D	279,–
ZING!	149,–
ZING KEYS	89,–
SILVER	279,–
TEST DRIVE	69,–
FIRE POWER	59,–
GARRISON	59,–
DETONATOR	49,–

ANIMATION

	DM
AEGIS Animator & Images	269
APPRENTICE 3d-Animator	498
APPRENTICE JUNIOR 3D-Animator	129
APPRENTICE Libraries: GEO	79
APPRENTICE Libraries: LETTERS	79
APPRENTICE Libraries: NUMBERS	69
DELUXE VIDEO II	198
PAGE FLIPPER deutsch	89
SILVER Ray Tracing Animator	279
VIDEOSCAPE 3D	279

BÜCHER

	DM
ADVANCED AMIGA BASIC	49
ADVANCED C PRIMER	69
AMIGA 1st COMPUTER	49
AMIGA APPLICATIONS	49
AMIGA BASIC IN&OUT BOOK	59
AMIGA C FOR BEGINNERS	49
AMIGA Jahrbuch 1987	10
AMIGA PROGRAMMERS GUIDE	49
AMIGA TRICKS&TIPS BOOK	49
AMIGA USER'S GUIDE	49
BEGINNERS GUIDE AMIGA	49
C PRIMER PLUS	69
C PROG. GRAPHICS BOOK	59
DOS Manual Book	59
ELEMENTARY AMIGA BASIC	49
FLIGHT SIM. ADV. BOOK	49
INSIDE AMIGA GRAPHICS	49
INSIDE THE AMIGA	69
KIDS AND THE AMIGA	49
MASTER AM DOS Buch	39
THE AMIGA BOOK	59
USING AMIGA DOS	49

DATENBANK

	DM
ACQUISITION	598
dBMan dBase III komp.	289
MICRO FICHE FILER	189
ORGANIZE database	189
SUPERBASE Personal	198

DEMOS

	DM
DEMO Acquisition	12
DEMO DIGI-VIEW	12
DEMO DisCovery	12
DEMO Dynamic CAD	12
DEMO EASYL Bilder	12
DEMO Gridirion	12
DEMO King of Chicago	12
DEMO Perfect Sound	12
DEMO POWER WINDOW & TxED	12
DEMO PRO WRITE & FLOW	12
DEMO VIZAWRITE Desktop	12
DEMO FUNKTION	12

DIVERSES

	DM
AEGIS DIGA!	149
BUSSINESS CARD MAKER	119
FLIPSIDE printer driver	98
FRED FISH Disks je	7
FUNKTION Graphenzeichner dt.	79
KICK START ELIMINATOR	198
PUBLIC DOMAIN Disks je	7

FESTPLATTEN

	DM
20 Meg. Hard Drive SCSI	1555
50 Meg. Hard Drive SCSI	2595
Harddisk 20 MB für A-2000/PC	998
Harddisk 20 MByte für A-2000	1498

GRAFIK

	DM
AEGIS DRAW	149
AEGIS DRAW PLUS	449
AEGIS IMAGES	79
AEGIS IMPACT	149
CALLIGRAPHER	209
DELUXE PRINT II	229
DELUXE PRINT & Data #1	139
DIGI PAINT	98
DYNAMIC CAD	898
FORMS IN FLIGHT 3D CAD Stereo	149
PRINTMASTER PLUS	98
PRISM #	98
SCULPT 3D Ray-Tracing	179
TV-TEXT 3D Textanim.	198

GRAFIKDATEN

	DM
AEGIS Artpack-1	59
ART GALLERY #1	59
ART GALLERY #2	59
DONALD DUCKS PLAYGROUND	69
GRAFIKBILDERDISK #1	12

KABEL

	DM
AMIGA 1000 Druckerkabel	29
AMIGA Bildschirmkabel Scart	29
CENTR.Adap. 500/2000 Bausatz	29
CENTRONICS Adapter A500/2000	49

KALKULATION

	DM
ANALYTIC ART	98
ANALYZE 2.0 spreadsheet	298
LOGISTIX	298
MAXIPLAN PLUS Lotus komp.	398

LAUFWERKE

	DM
Laufwerk 3½ Zoll extern	348
Laufwerk 3½ Zoll intern	328

MUSIK

	DM
AEGIS AUDIOMASTER	98
AEGIS SONIX Musikprogramm	149
DELUXE MUSIC Construction Set	198
DNA MUSIC	69
DYNAMIC DRUMS	139
ENCORE mouse/keyb/macro	89
EYE PLAY 2	69
FRACTAL MUSIC	69
FUTURE SOUND II	337
INSTANT MUSIC	98
MUSIC STUDENT	98
PERFECT SOUND mit Digitizer	169
PRO MIDI STUDIO	298
PROTEIN MUSIC	69
SOUND SCAPE Sampler	179
STUDIO MAGIC	98
THE MUSIC STUDIO	98
WAVEBUILDER Supersound	69
WAVEFORM EASEL	69

MUSIKDATEN

	DM
60's GREATEST (45 min.)	49
70's GREATEST (45 min.)	49
80's GREATEST (50 min.)	49
BARBERSHOP (65 min.)	49
BEATLES Part 1 (40 min.)	49
BEATLES Part 2 (40 min.)	49
BEETHOVEN (40 min.)	49
BILLY JOEL (65 min.)	49
BROADWAY THEMES (40 min.)	49
CHRISTMAS (50 min.)	49
CHURCH MUSIC (50 min.)	49
CLASSICAL #1 (40 min.)	49
CLASSICAL #2 (40 min.)	49
CLASSICAL #3 (35 min.)	49
CLASSICAL #4 (30 min.)	49
CLASSICAL #5 (50 min.)	49
COMPOSER DISK Volume I	69
COMPOSER DISK Volume II	69
COMPOSER DISK Volume III	69
COUNTRY #1 (45 min.)	49
COUNTRY #2 (50 min.)	49
GOLD&PLATINUM (60 min.)	49
INSTR. DISK B-3 Organ	69
INSTR. DISK Melotron	69
INSTR. DISK Steeldrums	69
IT'S ONLY ROCK'N'ROLL	69
KENNY ROGERS (45 min.)	49
MOVIE THEMES (40 min.)	49
NOSTALGIA (45 min.)	49
POLKA PARTY (40 min.)	49
R. RODGERS SONGBOOK	49
ROCK Part 1 (50 min.)	49
ROCK Part 2 (50 min.)	49
SYMPHONY JUKEBOX	49
SYMPHONY MUSIC VIDEO	49
TV THEMES (35 min.)	49

SIMULATION

	DM
ACHELON Flugsimulator	129
FLIGHTSIMULATOR II	98
GALILEO Planetarium	149
JET Simulator	98
Scenery Disk #7 od. #11	49
SUPER HUEY	59
SURGEON Operationssimulation	69

SPEICHER

	DM
COMSPEC 2 MByte RAM	998
Speicher 512KB & Uhr f. A-500	249
Speichererweiterung 2 MByte f. A2000	795

SPIELE

	DM
A MIND FOREVER VOYAGING	89
ADVENTURE CONSTRUCTION	69
ALIEN FIRES adventure	69
ALT. REALITY 'THE CITY'	89
ARAZOK'S TOMB adv.	59
ARCHON game	89
ARCTICFOX game	69
ARENA & BRATACASS	79
AUTO DUEL sim.	119
BALANCE OF POWER	79
BALLYHOO mystery	79
BARDS TALE adv.	89
BEAT IT	29
BLACK CAULDRONS	98
BLACK JACK game	69
BORROWED TIME game	69
BOWLING	79
BRIDGE 4.0	89
BUREACRACY comedy	79
CHALLENGER	25
CHAMPIONSHIP BASEBALL	69
CHAMPIONSHIP BASKETBALL	69
CHAMPIONSHIP FOOTBALL	79
CHAMPIONSHIP GOLF	79
CHESSMASTER 2000 Schach	79
CHESSMATE Schach	49
COMPUTER BASEBALL	68
D'BUDDY	129
DECIMAL DUNGEON educ.	98
DEEP SPACE	79
DEFENDERS OF THE CROWN	79
DEJA VU	79
DELUXE MAPS	59
DEMOLITION	25
DETONATOR Spitzensound	69
DISCOVERY TRIVIA game	98
DR. FRUIT	29
DR. XES	98
EARL WEAVER BASEBALL	98
ENCHANTER adv.	59
FAERY TALE ADVENTURE	89
FIGHTER COMMAND	89
FINAL TRIP	29
FIRE POWER game	59
GALACTIC INVASION game	49
GALAXY FIGHT	29
GARRISON	59
GOLDEN OLDIES	59
GOLDEN PYRAMID	69
GRAND SLAM TENNIS	69
GRIDIRION	119
GUILD OF THIEVES Adv.	79
GUNSHIP game	86
HACKER II game	79
HALLEY PROJECT	69
HEX game	69
HEX graphic/strategy	89
HITCHHIKERS GUIDE Adv.	59
HOLLYWOOD HIJINX	69
Hollywood Poker	49
JEWELS OF DARKNESS	69
KAMPFGRUPPE strategy	109
KARATE	69
KARATE KID II	74
KARATE KING	49
KING OF CHICAGO game	89
KINGS QUEST 1	89
KINGS QUEST 2	89
KINGS QUEST 3	89
KNIGHT ORC adv.	79
LAND OF LEGENDS adv.	89
LEADER BOARD (Golf Game)	69
LEADER BOARD TOURNAMENT	39
LEATHER GODESSES	79
LITTLE COMPUTER PEOPLE	59
LYBIANS IN SPACE	59
MAGICIANS DUNGEONS	79
MARBLE MADNESS	69
MIND BREAKER	29
MIND FOREVER	79
MINDSHADOW	69
MISSION ELEVATOR	49
MOONMIST	79
PACK BOY	29
PHALANX	25
PHANTASIE III Wrath of Nikad	98
POKER	79
PORTAL	79
RETURN TO ATLANTIS	109
ROADWAR 2000 game	98
ROGUE Adventure	68
SHADOWGATE	79
SHANGHAI	69
SHOOTING STAR	29
SILICON DREAMS	69
SINBAD game	79
SOLITAIRE	49
SPACE PORT	49
SPACE FIGHT	29
SPACE QUEST	98
STAR GLIDER game	79
STARFLEET I	89
STATIONFALL	79
STRIP POKER	69
TASS TIMES IN TONETOWN	79
TEMPLE OF ASPHAI	68
TENTH FRAME	79
TERRORPODS	59
TEST DRIVE	69
THE GRID	98
THE PAWN Adventure	79
TRINITY	79
TURBO car racing	49
ULTIMA III	89
UNINVITED	79
VADER	29
VIDEO VEGAS	59
WINTERGAMES	59
WISHBRINGER	39
WIZARDS CROWN	99
WORLD GAMES	59
ZORK Trilogy (I + II + III)	119

SPRACHEN

	DM
AC BASIC Compiler	398
AZTEC C Commercial	898
AZTEC C Developers	498
AZTEC C Personal	439
LATTICE C Compiler V 3.1	398
METACOMCO Lisp	398
METACOMCO Makro Assembler	169
METACOMCO Pascal (ISO)	189
MODULA-2 Commercial	598
MODULA-2 Developer	298
MODULA-2 Standard	198
PASCAL UCSD	198

SYSTEME

	DM
AMIGA 2000 Grundgerät	2395
AMIGA 500 Grundgerät	1149

TEXT

	DM
CITY DESK Desktop P.	298
DESKTOP ARTIST	79
DYNAMIC WORD	298
FAST FONTS	79
FLOW V1.02 Idea Proc.	149
ORDER Desktop organizer	89
PAGE-SETTER (Umlaute)	228
PAGESETTER LASERSCRIPT	89
PRO WRITE	198
PUBLISHER 1000	349
SCRIBBLE	189
TALKER wordprocessor	149
VIZAWRITE Desktop dtsch.	179
VIZAWRITE DESKTOP engl.	189
ZUMA FONTS Vol. 1	59
ZUMA FONTS Vol. 2	59
ZUMA FONTS Vol. 3	59

TOOLS

	DM
BUTCHER GRAPHIC-CONV. + EDITOR	69
CLI MATE Diskutility	69
DEMONSTRATOR	69
DisCovery Disk Editor deutsch	198
DISCOVERY-SHELL	119
DISK TO DISK	98
DOS EXPRESS	59
DOS to DOS	98
FLOPPY ACCELERATOR	59
GIZMOZ 2.0	128
GRABBIT	59
KEY-GENIE	119
MARAUDER II	69
METACOMCO Shell	98
METACOMCO Toolkit	89
MIRROR HACKER PACKAGE	98
MIRROR Kopierprogramm	98
POWER WINDOWS C-Libs & Help	149
QUICK NIBBLE Duplicator	79
THE EXPLORER Debugger	98
TxED Editor	79
WOW disk cache system	79
ZING KEYS	89
ZING! V1.02	149

VIDEO

	DM
DIGI-VIEW Digitizer	289

ZUBEHÖR

	DM
BASIC KEYBOARD OVERLAY	19
DOS KEYBOARD OVERLAY	19
EASYL 500 Zeichentablett	698
EASYL 1000 Zeichentablett	798
EASYL 2000 Zeichentablett	898
Erweiterungskit AT	1695
Erweiterungskit XT	1098
HOST ADAPTER SCSI	495
JITTER-RID Filterglas	29
Joystick CONTROLLER	14,95
Joystick J-1000 SUPERMICRO	19
Joystick STARFIGHTER HM	24
Joystick STARMASTER	19,80
Joystick TAC 2 Heavy Metal	29
MD-120 3½" Teakdiskbox	79
MD-60 3½" Teakdiskbox	59
MD-60L 3½" Teakdiskbox	79
MOUSE HIDE Leder Pad	29
MOUSE PAD (9½" ★11")	18
PAL Video Karte AMIGA 2000	198
PRO MIDI INTERFACE	98
REFERENCE PAK (5 Cards)	39
TIME SAVER dtsch.	178
TRACKBALL	89
TV Modulator für AMIGA	59

Preisänderungen und Irrtümer vorbehalten

amigaland
a. koppisch
hohenwaldstraße 26
6374 steinbach

Bestellservice:
06171/71846

Alle Preise sind Endpreise.
Mindestbestellwert DM 50,–.
Versand ausschließlich gegen Nachnahme oder Vorkasse.

Viel Fiction mit Floyd

AMIGA test

Gleich zweimal haben Sie die Gelegenheit, sich als Held in einem Abenteuer zu beweisen. Im Infocom-Adventure »Planetfall« müssen Sie »nur« einen Planeten retten, in »Stationfall« gleich das ganze Universum.

Sie würden so gerne ein Held sein, wenn Sie nur mal die Chance dafür bekämen. Deshalb haben Sie sich auch von der »Stellar Patrol« anwerben lassen. Die Werbeprospekte versprechen zwar eine Menge Spaß und Abenteuer für einen Kerl, wie Sie es sind. Aber die Realität sieht etwas anders aus.

Auf dem Raumkreuzer S.F.S. Feinstein haben Sie eine wenig ehrenvolle Aufgabe übernommen: den schmierigen Boden des achten Decks zu schrubben. Doch bevor Sie der übliche Routineschlaf übermannt, passiert etwas Unvorhergesehenes — die Feinstein löst sich in ihre Bestandteile auf. Plötzlich öffnet sich einladend die Tür zur Rettungskapsel. Ein kleiner Schritt, und schon sind Sie in einem Abenteuer, wie Sie es sich bisher nicht zu träumen wagten.

Die Rettungskapsel fliegt zum nächstgelegenen Planeten: Residia. Sie landen in einer hochtechnisierten Stadt und stellen schnell fest, daß sowohl die Stadt als auch der Planet nicht im allerbesten Zustand sind. Die Stadt ist verlassen und die meisten Einrichtungen zerstört, aber mit etwas Geschick reparierbar. Sie müssen schnell handeln, ansonsten ist der Planet dem Untergang geweiht.

Aber Sie sind nicht ganz allein. Schon bald stößt man auf den Allzweckroboter Floyd. Wer sich aber einen modernen, hyperintelligenten und vollpositronischen Robot vorstellt, wird ein wenig enttäuscht. Floyd ist anders als andere Roboter. Er hat ein ganz eigenes Ego. Er begleitet Sie fortwährend auf allen Wegen und versüßt so manche schlimme Situation durch einen (meist völlig unpassenden) Kommentar. Aber ohne ihn können Sie das Adventure nicht lösen.

Alle Puzzles in diesem Adventure sind streng logisch. Wenn man bei einem Puzzle partout nicht weiterkommt, kann man zu einem anderen Ort gehen und dort an einem anderen tüfteln. Planetfall ist nicht sehr schwer zu lösen; die Schwierigkeit wird von Infocom mit Standard angegeben.

Wie üblich lassen sich die Packungsbeilagen sehen: eine Broschüre über Geschichte und Wirken der Stellar Patrol, ein langer Brief, drei nette Postkarten von populären galaktischen Urlaubszielen und eine Ausweiskarte mit Magnetstreifen gibt es gratis.

Aber die Story geht weiter: Nach dem haarsträubenden Abenteuer sind Sie natürlich befördert worden. Als Leutnant erster Klasse haben Sie jetzt keine sonderlichen Schwierigkeiten mit Vorgesetzten, aber die neue Position ist alles andere als spannend. Sie sind jetzt der Obermotz der Bürohengste und langweilen sich schon wieder entsetzlich. Heute haben Sie einen äußerst erregenden Auftrag bekommen. Sie sollen einen Antrag für einen Antrag auf einen Formularordner für Anträge von der nahegelegenen Raumstation abholen. Also schnappen Sie sich einen Roboter und fliegen zur Station. Sie ahnen es sicher: Der Roboter enpuppt sich Ihr treuer Freund Floyd, der ebenfalls in die Dienste der Stellar Patrol trat (Bild 1).

Dort angekommen, merken Sie und Floyd schnell, daß da so einiges nicht stimmt. Die Station ist völlig verlassen, und alle mechanischen Geräte verhalten sich keinenfalls so, wie sie sollten. Schon bald stoßen Sie auf ein seltsames Schiff, in dem ein totes Alien liegt. Ob es die Ursache für den desolaten Zustand der Station ist? Sie sollten sich beeilen, die Gründe für das seltsame Verhalten herauszufinden, denn sogar Ihr Begleiter Floyd beginnt, sich äußerst merkwürdig zu verhalten (Bild 2).

Auch hier bekommt man als Käufer eine Menge Schnick-Schnack für sein Geld, ohne den man das Spiel nicht lösen kann. Drei Formulare, ein Aufnäher der Stellar Patrol und die gesamten Blaupausen der Station liegen in der Packung.

Beide Adventures sind reine Textadventures. Wer sich also von Planet- oder Stationfall tolle Grafiken erwartet, wird enttäuscht sein. Dafür bekommen Sie zwei flott geschriebene Geschichten mit viel Humor und knackigen Puzzles.

(Anatol Locker/jk)

```
Workshop                                                Score: 5/6416
The auto-door opens as you approach, and whisks shut behind you...

Workshop
   This is a fairly large space where much of the maintenance work for the
station takes place. Some repair work for visiting ships is also done here.
The room is filled with all sorts of repair and fabrication machinery. A
door leads south, and there's an opening to the north.
   You can see a twenty-ohm bedistor here.
   Floyd dashes into view, followed by a slightly older-looking robot.
"Look, Floyd found a new friend," Floyd says with unbounded exuberance.
   "I am quite surprised to discover you here," says the robot. "I have not
seen a soul for a day now, perhaps more. But look, here I am forgetting my
manners again. I am known as Plato to the humans on this station, and I am
most gratified to make your acquaintance."

>examine plato
Plato is slightly taller than Floyd; in addition, he seems to be wiser and
older. Overall, he leaves you with the impression that he's somewhat of a
bookworm.
   Floyd and Plato discuss their favorite battery recharging techniques.

>
```

Bild 1. Der Roboter »Floyd« hilft Ihnen bei Problemen

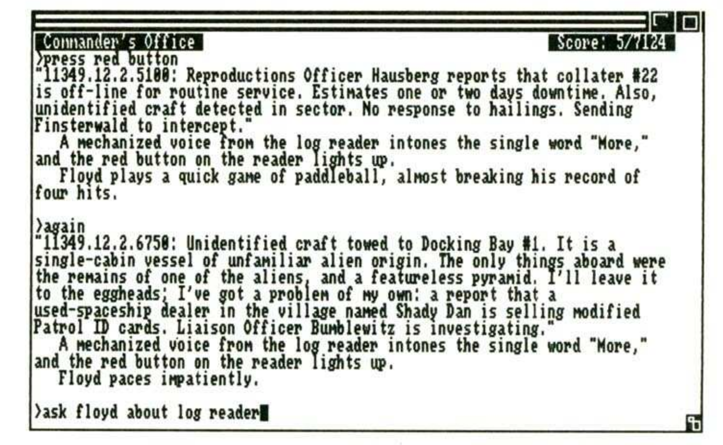
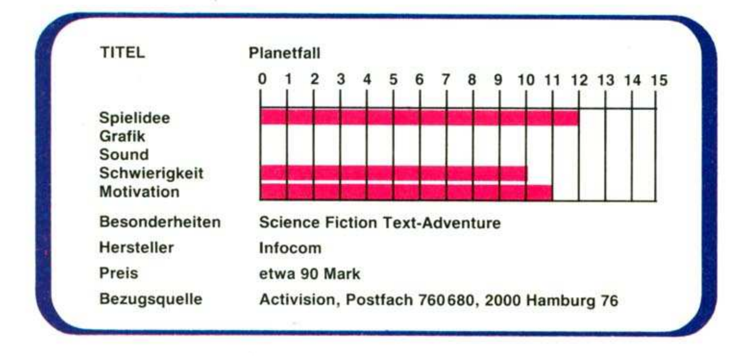
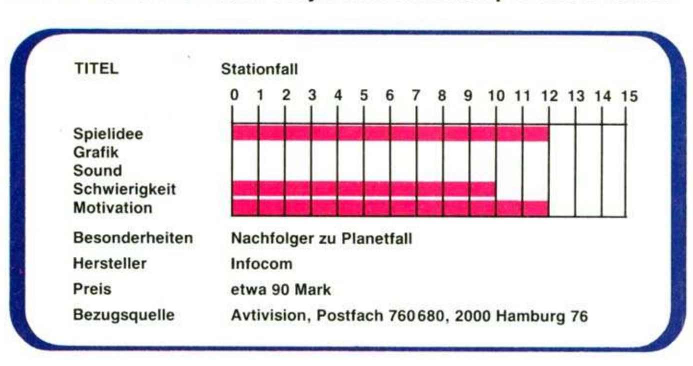


Bild 2. Manchmal kann Floyd aber auch empfindlich stören



TITEL	Planetfall
Spielidee	12
Grafik	
Sound	
Schwierigkeit	10
Motivation	11
Besonderheiten	Science Fiction Text-Adventure
Hersteller	Infocom
Preis	etwa 90 Mark
Bezugsquelle	Activision, Postfach 760680, 2000 Hamburg 76



TITEL	Stationfall
Spielidee	12
Grafik	
Sound	
Schwierigkeit	10
Motivation	12
Besonderheiten	Nachfolger zu Planetfall
Hersteller	Infocom
Preis	etwa 90 Mark
Bezugsquelle	Avtivision, Postfach 760680, 2000 Hamburg 76

Spannende Joystick-Action

AMIGA test

Die Spielhallen-Freaks unter den Amiga-Besitzern kommen langsam auf Ihre Kosten. Wenn schon nicht die Original-Spiele umgesetzt werden, dann hält man sich eben an die ähnlichen Clones.

Die Software-Produzenten aus England und Amerika scheinen die Amiga-Besitzer im Moment etwas im Stich zu lassen. Zum Glück gibt es noch die deutschen Programmierer, deren erklärter Lieblingscomputer anscheinend der Amiga ist. Vor kurzem wurden wieder zwei neue Action-Spiele veröffentlicht, die beide auf bekannten Spielautomaten basieren.

»Garrison« ist eine gut gemachte Variante des erfolgreichen Münzenschluckers »Gauntlet« (Bild 1). Eine Umsetzung des Originals ist zwar angekündigt, aber leider immer noch nicht erhältlich. Wer nicht mehr so lange warten will, kann sich mit der sehr guten Kopie die Zeit vertreiben.

Ein oder zwei Spieler gleichzeitig schießen sich durch Hunderte von verschiedenen Labyrinthen. In den Gängen stehen Generatoren, die unterschiedliche Monstertypen ausspucken. Diese widerlichen Kreaturen haben es natürlich auf Sie abgesehen und wollen Ihnen wertvolle Energie stehlen. In den Labyrinthen finden sich ab und zu Mahlzeiten, die den Energievorrat wieder aufstocken. Weitere Gefahren lauern unterwegs auf den Spieler. Unsichtbare Mauern, bewegliche Wände und Morast sind nur ein paar von vielen Gefahren, die dem wackeren Krieger das Leben schwer machen.

Zu Beginn darf man sich eine von fünf Spielfiguren aussuchen. Diese weisen unterschiedliche Stärken auf (Geschwindigkeit, Nahkampf, magische Kraft, Rüstung, Schußgeschwindigkeit und Schußstärke). Deren Wirkung kann man während des Spiels durch Einsammeln von Extras verbessern. Für jede Spielfigur wird eigens eine High-Score-Liste auf Diskette angelegt. Garrison nützt sogar die PAL-Auflösung des Amiga aus.

Im Vergleich zum Original ist das Scrolling und die Bewegung der Figuren bei Garrison erheblich langsamer, was dem Spielspaß etwas schadet. Das soll aber nicht heißen, daß es schlecht wäre. Vor allem zu zweit ist Garrison recht unterhaltsam. Da es insgesamt 128 verschiedene Level gibt, ist für Abwechslung gesorgt.

Die Programmierer von »Mission Elevator« haben sich ebenfalls von einem Spielautomaten inspirieren lassen (Bild 2). Das Vorbild, »Elevator Action«, war vor Jahren in den Spielhallen dicht umlagert.

In einem großen Hotel, das dem FBI als Zentrale dient, wurde von ausländischen Geheimagenten eine Zeitbombe versteckt. Einem gewitzten Computer-Experten ist die Erkenntnis zu verdanken, daß sich die Bombe im 62. Stockwerk befindet. Kurz bevor er den Stopp-Code eingeben konnte, wurde er jedoch von den Schurken gefaßt und in den Keller verschleppt. Auf dem Weg nach unten gelang es ihm, an verschiedenen Stellen Hinweise zu verstecken, mit deren Hilfe man den Code ermitteln kann. Ihnen wurde die verantwortungsvolle Aufgabe übertragen, in das Hotel einzudringen und die gefährliche Bombe zu entschärfen.

Mit einer Pistole bewaffnet, machen Sie sich auf den Weg. Innerhalb von jeweils acht Stockwerken kann man sich mit Hilfe von Fahrstühlen bewegen. Um in die höheren Stockwerke zu gelangen, muß ein passender Schlüssel gefunden werden. Sie sollten jede Hoteleinrichtung, ob Wandbild oder Sessel, genau unter die Lupe nehmen. Beinahe unentwegt werden Sie von finsteren Typen beschossen, die in höheren Stockwerken immer zahlreicher erscheinen. Nette Einlagen wie eine Pokerpartie oder eine Wette mit dem Barkeeper lockern den Spieleralltag merklich auf.

Mission Elevator spielt sich gut. Die vielen kleinen Gags am Rande machen sich positiv bemerkbar. Das vertikale Scrolling ist hervorragend, und die Animation der Personen recht ordentlich. Die digitalisierten Soundeffekte sind ebenfalls gelungen. Alles in allem ein Spiel, das zwar nicht besonders viel Abwechslung bietet, aber trotzdem deutlich über dem Durchschnitt liegt.

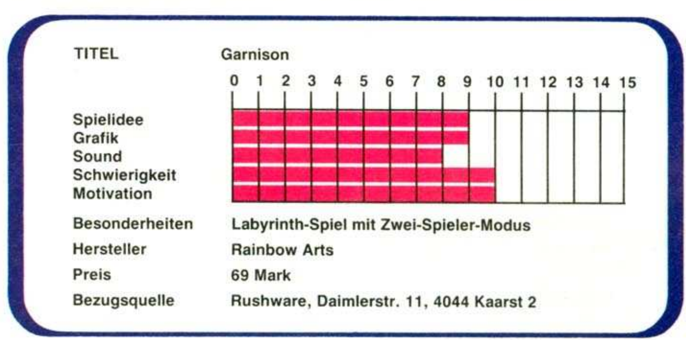
(Martin Gaksch/jk)

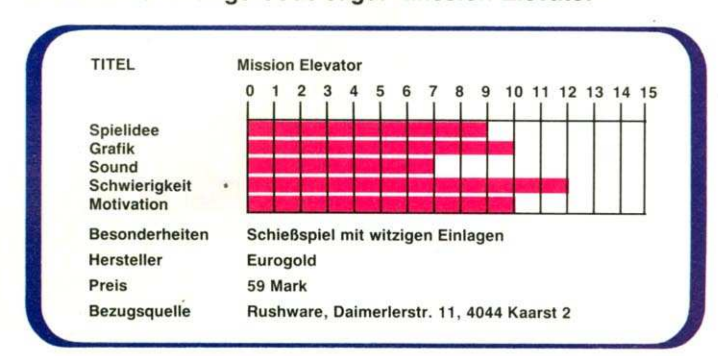


Bild 1. Mit zwei Spielern sehr fesselnd: »Garrison«



Bild 2. Eine knifflige Code-Jagd: »Mission Elevator«





Klassische Umsetzungen

AMIGA test

Programme, die auf Spielhallenautomaten laufen, werden gerne auf andere Computer umgesetzt. Mit »City Defence« und »Fortress Underground« liegen jetzt zwei bekannte Spielprinzipien vor.

Deutsche Programmierer haben den Amiga als Spielecomputer entdeckt. Zunehmend tauchen neue Versionen bekannter Spiele für den Amiga auf. Dabei hält man sich allerdings nicht nur an Umsetzungen von Spielhallenautomaten. Auch mit Spielen, die auf anderen Computern sehr guten Anklang gefunden haben, versucht man inzwischen, erfolgreich zu sein.

Wenden wir uns als erstes »City Defence« zu (Bild 1). Vom Spielprinzip her ist dies fast eine 1:1-Umsetzung des Ende der siebziger Jahre sehr beliebten Spielautomaten. Die Idee, die dahinter steckt, ist sehr einfach erklärbar. Es gilt einem Raketenangriff aus dem Weltall standzuhalten. Man besitzt dazu zwei Abwehrbasen, die jeweils mit 15 Abwehrraketen zur Verteidigung ausgerüstet sind. Um diese beiden Basen herum liegen sechs Städte, die vor der Vernichtung durch die heransausenden Gegner bewahrt werden müssen. So einfach das Ganze auch klingen mag, während des Spiels ist vor allem in höheren Spielstufen auf dem Bildschirm die Hölle los. Im ersten Level kommen die feindlichen Raketen noch ganz gemächlich auf ihre Ziele zugekrochen. Für jede abgeschossene Rakete erhält man natürlich Punkte. Am Ende einer jeden Spielstufe werden jedoch die nicht verbrauchten Verteidigungsraketen und alle verbliebenen Städte in Bonuspunkte umgewandelt. Wenn sich allerdings die Geschwindigkeit des Angriffs steigert, ist man später nur noch darauf bedacht, überhaupt noch eine einzige Stadt vor dem Angriff der fiesen Aliens zu bewahren. Als besondere Feinde fliegen dann auch manchmal noch Sonden oder Raumminen über den Bildschirm, mit denen man Extrapunkte machen kann. Das Knifflige an der ganzen Situation ist jedoch, daß die eigenen Raketen vom Abschußpunkt der Basis bis zum Ziel eine gewisse Weile brauchen. Daher erfordert das Spiel viel Timing, Fingerspitzengefühl und natürlich auch Übung. Gesteuert wird das Fadenkreuz mit der Maus. City Defence darf auch mit zwei Spielern abwechselnd gespielt werden. Die speicherbare High-Score-Liste sowie die sauber digitalisierte Titelmelodie verstärken den guten Gesamteindruck. Grafik und Sound sind insgesamt gesehen nicht sehr aufwendig, aber effektiv gestaltet. Die Spielidee und vor allem die Schnelligkeit des Spiels wird dadurch hervorragend unterstützt. Man sollte gewarnt sein: City Defence kann auf ganz subtile Weise süchtig machen.

Ums Überleben, allerdings auf etwas andere Weise, geht es auch bei »Fortress Underground« (Bild 2). Hier übernehmen Sie die Rolle eines Hubschrauberpiloten, der in einer unterirdischen Höhle ein feindliches Kraftwerk aufspüren und vernichten soll. Insgesamt stehen Ihnen sechs Kampfhubschrauber für diese Mission zur Verfügung. Da Sie ständig Treibstoff verbrauchen, sind an einigen Stellen des Höhlenlabyrinths »Fuel-Depots« angebracht, die man immer wieder anfliegen muß. Damit das Ganze nicht zu schwierig ist, sind zusätzlich Punkte mit »Stay Here« markiert, von denen man immer wieder starten kann, wenn sie einmal angeflogen wurden. So braucht man das Spiel nicht ständig wieder von vorne beginnen. Die Grafik reizt den Amiga nicht richtig aus, ist aber recht nett gemacht. Manko ist jedoch, daß der Hubschraubersound während der gesamten Mission aus den Lautsprechern dröhnt und sich nicht abstellen läßt. Er geht schon nach kurzer Zeit fürchterlich auf die Nerven. Es gibt keinen Zwei-Spieler-Modus und auch keine High-Score-Liste. Durch die Enge des Labyrinths, die schwierige Steuerung und die schießwütigen Angreiferuntertassen erfordert das Spiel recht viel Übung. Aus dem recht einfachen Spielprinzip hätte man mehr machen können; so ist nur ein müder Abklatsch herausgekommen. (jk)

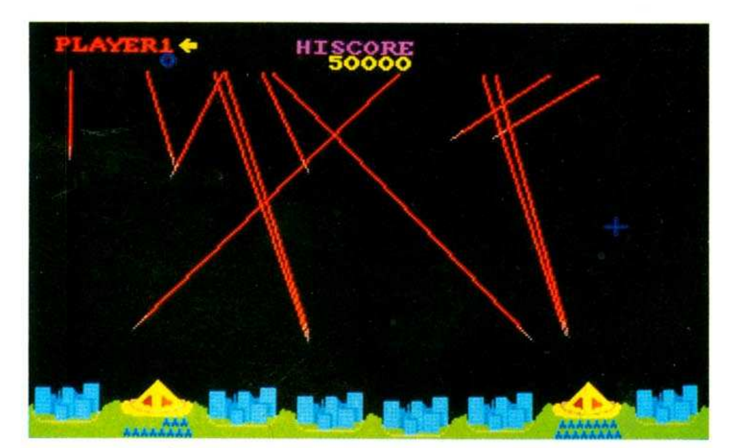
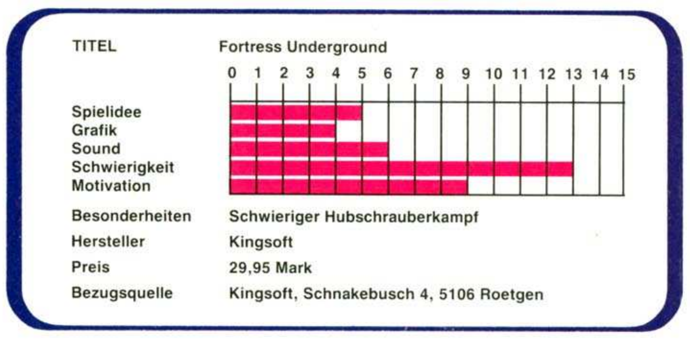


Bild 1. Städte verteidigen, erfrischend einfach umgesetzt



Bild 2. Hakeliger Kampf gegen die Festung im Untergrund





Domino perfekt!

Es gibt viele, die gerne spielen. Für manche Spiele ist aber ein Partner unerläßlich. Und wenn sonst niemand da ist, spielt eben der Computer.

Ein Spiel, das man auch gegen den Computer spielen kann, ist Domino. Bei der vorliegenden Version besticht vor allem anderen die grafische Aufmachung (siehe Bild). Zum Beispiel wirken die Steine durch den Schatten sehr plastisch.

Aber auch die Bedienung ist sehr einfach, da alles mit der Maus erledigt wird. Vor allem werden keine Pull-Down-Menüs, sondern Gadgets (Schalter) verwendet, die alle sichtbar sind. Die Sprachausgabe des Programms ist deutsch und recht gut verständlich. Allerdings muß hierfür auf der Bootdiskette im Unterverzeichnis »devs« die Datei »narrator.device« vorhanden sein.

Das Programm wirkt insgesamt sehr gut und wurde aus diesen Gründen auch von uns zum Programm des Monats gewählt.

Nach dem Starten des Programms werden Sie zunächst vom Programm begrüßt. Nach einer kurzen Wartezeit erscheint dann die Spielfläche. In der Mitte des Feldes liegt schon der Stein mit der höchsten Punktzahl, welcher entweder vom Spieler oder vom Computer genommen wurde.

Am oberen Bildschirmrand sind fünf Schalter zu sehen:

— Patt:

Sollte einmal das Spiel nicht aufgehen, kann man durch Anklicken dieses Schalters das Spiel beenden. Nach der bis dahin erreichten Punktzahl wird dann der Sieger berechnet.

— Ende:

Durch Anwählen dieses Schalters wird das Programm beendet.

— Info:

Nach Anklicken des Gadgets wird eine Kurzanleitung angezeigt. Durch einfaches Klicken mit der linken Maustaste wird das Fenster wieder entfernt und das Spiel geht weiter.

— Weiter:

Wenn Sie nicht anlegen können, müssen Sie normalerweise einen neuen ziehen. Durch Klick auf diesen Schalter wird jedoch die Kontrolle einfach an den Computer übergeben.

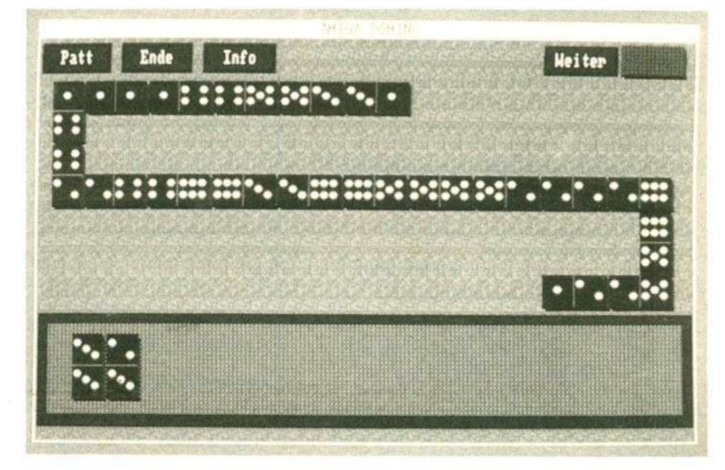
Das letzte Feld ganz rechts stellt den Stapel der noch nicht benutzten Steine dar. Von hier ziehen Sie neue Steine.

Wie oben schon erwähnt, liegt gleich bei Spielbeginn der Stein mit der höchsten Augenzahl in der Mitte des Spielfeldes.

Um einen Stein an die Schlange anzulegen, muß man zunächst einen der zwei Steine am Ende der Schlange anklicken. Dabei ist darauf zu achten, daß der Mauszeiger über der am Ende liegenden Hälfte des Steins steht. Hat man dies getan, wird der Stein rot eingefärbt. Nun können Sie aus Ihrem Steinvorrat in dem roten Kasten unten auf dem Bildschirm einen Stein auswählen. Haben Sie einen passenden Stein gewählt, wird dieser angelegt.

Auch bei Domino kann man durch Taktieren den Gegner öfter besiegen, was der Computer dann auch etwas zerknirscht zugibt.

(Stefan Boldorf/rb)



Das Spielfeld von »Domino« mit den fünf Schaltern

Programmname:	Domino
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2

```
 1 VQ0 ' Programm Amiga Domino
 2 2n  ' von Stefan G. Boldorf in July/August 1987
 3 qc2   DEFINT a-z
 4 HG    a$ = "                         AMIGA DOMINO "
 5 rf    SCREEN 2,640,256,3,2
 6 LV    WINDOW 1,a$,(0,0)-(631,200),16,2
 7 Tv    FOR t = 0 TO 7
 8 BK4     PALETTE t,0,0,0
 9 wN2   NEXT t
10 xn    GOSUB Vorspann
11 Oa    DIM Eins%(200), Zwei%(200), Drei%(200)
12 dA    DIM Vier%(200), Fuenf%(200), Sechs%(200)
13 NR    DIM Zahl%(200), Weiter%(200), Patt%(200)
14 No    DIM Tile$(36), ASF%(200), INF%(200)
15 om    DIM Stapel$(1,26), Verschm(36), Gunst(36)
16 oD    DIM Anzahl(1), AKTT(1)
17 Kh    DIM PX(1), PY(1), ADY(1), ADX(1), ZZ(1)
18 Uw    DIM KAX(4), KBX(4), KAY(4), KBY(4)
19 PQ    DIM MusterA%(1), MusterB%(1), Tisch%(31), Laber%(8)
20 wj    Laber%(0) = 90 : Laber%(1) = 1 : Laber%(2) = 130 : Laber%(
         3) = 0
21 Kh    Laber%(4) = 20000 : Laber%(5) = 64 : Laber%(6) = 5 : Laber
         %(7) = 0
22 Ta    Laber%(8) = 0
23 07    IchBinDran$ = "AYN EHN   M OHM EHN T   B IHT T EH."
24 Os    Welcome$ = "W IHL K AOMAEN   Z UHM   D OHM IHN OH ."
25 ux    Lament$  = "W IY   S OHL L   IH/C   D AHS   M AH/CN ?"
26 Mp    Tschues$ = "AWF   WIYD EHR S EHEHN ."
27 3L    NMM$     = "D AHS   G EHD   N IH/CT   M EHR ."
28 5Q    OK$      = "OH    K AEIH."
29 Z5    IKNM$    = "IH/C  K AHN    N IH/CT   M EHR ."
30 qq    NewGame$ = "AYN   N OYEHS   S P IYL ?"
31 x3    MusterA%(0) = &HFFFF : MusterA(1) = &HFFFF
```

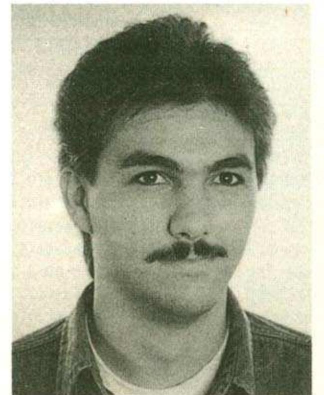
Listing. »Domino«: Bitte mit dem Checksummer eingeben

Jeden Monat 2000 DM GEWINNEN

Das Programm des Monats stammt diesmal von Stefan Boldorf. Den ersten Kontakt zu Computern hatte er 1983 in seiner Schule. Durch den dort vorhandenen Apple IIe angeregt, kaufte er sich einen C 64. Über Basic gelangte er schließlich zur Assemblerprogrammierung.

Im Dezember 1986 stieg er dann auf den Amiga 1000 um, von dem er schon immer beeindruckt war. Im Juni dieses Jahres wurde der Amiga 1000 in einen Amiga 2000 umgetauscht.

Eigentlich wollte der Autor nur einen Computerspieler schreiben, aber das Programm wurde immer größer, und so entstand schließlich Domino.



Zur Zeit lernt der Autor C, um den Amiga besser ausnutzen zu können.

```
32 Z7    MusterB%(0) = &H5555 : MusterB(1) = &H8888
33 1w    SAY Welcome$,Laber%
34 js    RANDOMIZE TIMER
35 q20 GOTO Main
36 KJ  GetTile:
37 p22   IF dran = 0 THEN SAY "IH/C  M UHS   (TS IY IY) (EH EHN).",
         Laber%
38 m2    IF TS < 37 AND Anzahl(dran) < 16 THEN
39 aM4     Anzahl (dran) = Anzahl(dran) + 1
40 jf      Stapel$(dran,Anzahl(dran)) = Tile$(TS)
41 jy      TS = TS + 1
42 rh      NixMehr = 1
43 L02   ELSEIF dran = 1 THEN
44 2R4     SAY NMM$,Laber%
45 K32   ELSE
46 ct4     SAY IKNM$,Laber% : NixMehr = -1
47 ha2   END IF
48 8k0 RETURN
49 88  Cleanup:
50 Pn2   FOR t = AKTT(dran) TO Anzahl(dran)
51 LB4     Stapel$(dran,t) = Stapel$(dran,t+1)
52 d42   NEXT t
53 1b    Anzahl(dran) = Anzahl(dran) -1
54 EqO RETURN
55 cX  MakeTiles:
56 Gu2   FOR a = 0 TO 5
57 X94     FOR b = 1 TO 6
58 r46       Tile$(a*6+b) = RIGHT$(STR$(a+1),1)+RIGHT$(STR$(b),1)
59 AJ4     NEXT b
60 9H2   NEXT a
61 LxO RETURN
62 Ld  ShuffleTiles:
63 tF2   SWAP Tile$(36),Tile$(INT(RND(1)*35)+1)
64 Ou    SWAP Tile$(1),Tile$(INT(RND(1)*35)+1)
65 4f    FOR t = 0 TO 200
66 if4     SWAP Tile$(INT(RND(1)*35)+1),Tile$(INT(RND(1)*35)+1)
67 sJ2   NEXT t
68 S40 RETURN
69 ww  SpreadTiles:
70 Xy2   Max(1) = 0 : Max(0) = 0
71 ez    FOR t = 1 TO 10
72 Gv4     Stapel$(0,t) = Tile$(t)
73 hn      Stapel$(1,t) = Tile$(t+10)
74 jd      Tile$(t) = "XX" : Tile$(t+10) = "XX"
75 OR2     FOR tt = 0 TO 1
76 6k4       a = VAL(LEFT$(Stapel$(tt,t),1))
77 I2        b = VAL(RIGHT$(Stapel$(tt,t),1))
78 re        IF a + b > Max(tt) THEN
79 796         Max(tt) = VAL(LEFT$(Stapel$(tt,t),1))+VAL(RIGHT$(Stape
               l$(tt,t),1))
80 zi          AKTT(tt) = t
81 F84       END IF
82 By      NEXT tt
83 8Z2   NEXT t
84 BL    Anzahl(0) = 10 : Anzahl(1) = 10  : TS = 21
85 JLO RETURN
86 A7  DisplayTiles:
87 Tt2   LINE (0,140)-(632,194),2,bf
88 vx    AREA (10,145)
89 aP    AREA (621,145)
90 zv    AREA (621,188)
91 FO    AREA (10,188)
92 gJ    PATTERN &HFFFF,MusterB%
93 Jc    COLOR 2,0
94 Zd    AREAFILL 0
95 gI    PATTERN &HFFFF,MusterA%
96 p5    CALL Stein (560,1,0,0,"V")
97 EQ    AREA (560,1)
98 2B    AREA (620,1)
99 G1    AREA (620,16)
100 U2   AREA (560,16)
101 OS   COLOR 7,4
102 qT   PATTERN &HFFFF,MusterB%
103 im   AREAFILL 0
104 pR   PATTERN &HFFFF,MusterA%
105 4S   FOR t = 1 TO Anzahl(1)
106 1m4    LinkerWert = VAL(LEFT$(Stapel$(1,t),1))
107 WS     RechterWert = VAL(RIGHT$(Stapel$(1,t),1))
108 Sh     CALL Stein(t*35,150,LinkerWert,RechterWert,"H" )
109 Yz2  NEXT t
110 8k0 RETURN
111 NZ InitAll:
112 I12  LINE(0,0)-(30,15),7,bf
113 m9   FOR t = 0 TO 2
114 V04    CIRCLE (5+t*9,4),4,4
115 Aq     CIRCLE (5+t*9,4),3,1
116 JK     PAINT (5+t*9,4),1,1
117 Lw     CIRCLE (5+t*9,10),4,4
118 Bi     CIRCLE (5+t*9,10),3,1
119 1J     PAINT (5+t*9,10),1,1
120 jA2  NEXT t
121 7A   GET (0,0)-(30,14),Sechs%
122 DA   LINE (0,0)-(30,15),7,bf
123 rD   FOR t = 0 TO 1
124 Yb4    CIRCLE (5+t*18,4),4,4
125 ON     CIRCLE (5+t*18,4),3,1
126 VW     PAINT (5+t*18,4),1,1
127 U6     CIRCLE (5+t*18,10),4,4
128 Gt     CIRCLE (5+t*18,10),3,1
129 UM     PAINT (5+t*18,10),1,1
130 tK2  NEXT t
131 1i   GET (0,0)-(30,14),Vier%
132 Ia   CIRCLE (5+9,7),4,4
133 5J   CIRCLE (5+9,7),3,1
134 eJ   PAINT (5+9,7),1,1
135 M5   GET (0,0)-(30,14),Fuenf%
136 RO   LINE (0,0)-(30,15),7,bf
137 Nf   CIRCLE (5+9,7),4,4
138 AO   CIRCLE (5+9,7),3,1
139 JO   PAINT (5+9,7),1,1
140 8i   GET (0,0)-(30,14),Eins%
141 WT   LINE (0,0)-(30,15),7,bf
142 Y8   CIRCLE (5,4),4,4
143 Ou   CIRCLE (5,4),3,1
144 5W   PAINT (5,4),1,1
145 hT   CIRCLE (5+18,10),4,4
146 MJ   CIRCLE (5+18,10),3,1
147 9W   PAINT (5+18,10),1,1
148 XF   GET (0,0)-(30,14),Zwei%
149 Zr   CIRCLE (5+9,7),4,4
150 Ma   CIRCLE (5+9,7),3,1
151 va   PAINT (5+9,7),1,1
152 L1   GET (0,0)-(30,14),Drei%
153 17   CLS
154 h0   LOCATE 1,1 : COLOR 1,2 : PRINT " Weiter "
155 Kt   GET (0,0)-(63,7),Weiter%
156 Mv   LOCATE 1,1 : COLOR 1,2 : PRINT "  Patt  "
157 r6   GET (0,0)-(63,7),Patt%
158 Oz   LOCATE 1,1 : COLOR 1,2 : PRINT "  Ende  "
159 Yw   GET (0,0)-(63,7),ASF%
160 A5   LOCATE 1,1 : COLOR 1,2 : PRINT "  Info  "
161 h8   GET (0,0)-(63,7),INF%
162 Nf   COLOR 1,0
163 zb0 RETURN
164 na ComputerPlayer:
165 od2  NixMehr = 0
166 Rv   SAY IchBinDran$,Laber%
167 rz   IF SC = 0 THEN GH = 1 : ELSE GH = 0
168 RD   Verschm(VAL(Wert$(GH))) =  Verschm(VAL(Wert$(GH))) + 1
169 EKO WarWohlNix:
170 7N2  IF NixMehr = -1 THEN LOOP
171 Qm   Allmax = -1
172 Oq   dran = 0
173 Ot   FOR a = 1 TO 6 : Prefer(a) = -1 : NEXT a
174 9W   FOR t = 1 TO Anzahl(0)
175 Je4    AugenA = VAL(RIGHT$(Stapel$(0,t),1))
176 TU     AugenB = VAL(LEFT$(Stapel$(0,t),1))
177 QA     Prefer(AugenA) = Prefer(AugenA) +1
178 bI     Prefer(AugenB) = Prefer(AugenB) +1
179 g72  NEXT t
180 Fc   FOR t = 1 TO Anzahl(0)
181 eB4    Gunst (t) = 0
182 Qw     a = LEFT$(Stapel$(0,t),1) = Wert$(0)
183 Z2     b = LEFT$(Stapel$(0,t),1) = Wert$(1)
184 Uc     c = RIGHT$(Stapel$(0,t),1) = Wert$(0)
185 ZJ     d = RIGHT$(Stapel$(0,t),1) = Wert$(1)
186 PH     IF a OR b OR c OR d THEN Gunst(t) = 500
187 Fb     Gunst(t) = Gunst(t) + Prefer(VAL(LEFT$(Stapel$(0,t),1)))
188 Iz     Gunst(t) = Gunst(t) + Prefer(VAL(RIGHT$(Stapel$(0,t),1))
           )
189 HJ     Gunst(t) = Gunst(t) + Verschm(VAL(RIGHT$(Stapel$(0,t),1)
           ))
190 vZ     Gunst(t) = Gunst(t) + Verschm(VAL(LEFT$(Stapel$(0,t),1))
           )
191 G4     IF Gunst(t) > 499 AND Gunst(t) > Allmax THEN
192 I06      Allmax = Gunst(t)
193 ty       ak = t
```

Listing. »Domino«: Bitte mit dem Checksummer (Seite 66) eingeben.

```
194 4x4     END IF
195 wN2   NEXT t
196 yX    IF Allmax < 499 THEN GOSUB GetTile : GOTO WarWohlNix
197 3P    FOR t = 0 TO 1
198 kJ4     IF LEFT$(Stapel$(0,ak),1) = Wert$(t) THEN SC = t : SideT
            aken = 1
199 HD      IF RIGHT$(Stapel$(0,ak),1) = Wert$(t) THEN SC = t : Side
            Taken = 0
200 1S2   NEXT t
201 rJ    dran = 0
202 1I0 GOTO PutTileOnScreen
203 Ov  Main:
204 Mt2   GOSUB InitAll
205 Y1    FOR t = 1 TO 6 : Verschm(t) = 0 : NEXT t
206 OI    FOR t = 0 TO 31
207 kw4     Tisch%(t) = INT ((RND(1) * 65535& )+ &H8000)
208 9a2   NEXT t
209 Zy    GOSUB MakeTiles
210 JI    GOSUB ShuffleTiles
211 iv    PATTERN &HFFFF,Tisch%
212 Nh    COLOR 0,3
213 pF    AREA (0,0) :AREA (631,0) : AREA (631,200) :AREA (0,200)
214 VZ    AREAFILL 0
215 gg    SAY NewGame$,Laber%
216 dF    PATTERN &HFFFF,MusterA%
217 GY    COLOR 1,0
218 HI    LINE (487,2)-(556,17),7,bf
219 DA    LINE (490,3)-(554,15),2,bf
220 qo    PUT ( 490,6),Weiter%,PSET
221 NT    LINE (1,2)-(70,17),7,bf
222 cB    LINE (4,4)-(67,15),2,bf
223 Fl    PUT ( 4,6),Patt%,PSET
224 K5    LINE (81,2)-(150,17),7,bf
225 SC    LINE (84,4)-(147,15),2,bf
226 t0    PUT (84,6),ASF%,PSET
227 1v    LINE (161,2)-(230,17),7,bf
228 8G    LINE (164,4)-(227,15),2,bf
229 3L    PUT (164,6),INF%,PSET
230 0g    PALETTE 0,0,.5,0
231 Zb    PALETTE 1,1,1,1
232 2k    PALETTE 2,.5,0,0
233 Dv    PALETTE 3,0,.4,0
234 Q7    PALETTE 4,.3,.3,.3
235 lt    PALETTE 5,0,0,0
236 PA    PALETTE 6,.2,.2,.2
237 1M    PALETTE 7,.13,.13,.13
238 Nw    GOSUB SpreadTiles
239 hl    IF Max(0) > Max(1) THEN dran =  0 : ELSE  dran = 1
240 BQ    LinkerWert = VAL(LEFT$(Stapel$(dran,AKTT(dran)),1))
241 ca    RechterWert = VAL(RIGHT$(Stapel$(dran,AKTT(dran)),1))
242 ns    CALL Stein(298,70,LinkerWert,RechterWert,"V" )
243 XC    Wert$(0) = RIGHT$(Stapel$(dran,AKTT(dran)),1)
244 r7    Wert$(1) = LEFT$(Stapel$(dran,AKTT(dran)),1)
245 a5    GOSUB Cleanup
246 zJ    GOSUB DisplayTiles
247 sq    PX(0) = 298 : PY(0) = 70 : ADX(0) = 63 : ADY(0) = 0 : ZZ(0
          ) = 0
248 B7    ZR$(0) = "V" : Richtung (0) = 2
249 cs    PX(1) = 298 : PY(1) = 70 : ADX(1) =-63 : ADY(1) = 0 : ZZ(1
          ) = 0
250 PK    ZR$(1) = "V" : Richtung (1) = 4
251 zq    KAX(1) = 0  : KBX(1) = 30 : KAY(1) = 0  : KBY(1) = 15
252 LI    KAX(2) = 30 : KBX(2) = 60 : KAY(2) = 0  : KBY(2) = 15
253 8G    KAX(3) = 0  : KBX(3) = 30 : KAY(3) = 15 : KBY(3) = 30
254 ui    KAX(4) = 0  : KBX(4) = 30 : KAY(4) = 0  : KBY(4) = 15
255 Rc    SC = 4
256 Jl    IF dran = 1 THEN ComputerPlayer
257 GaO LOOP:
258 qJ2   dran = 1
259 Di    CALL Maus(MX,MY)
260 S3    SCE = -1
261 y8    IF MX > 560 AND MX < 620 AND MY > 1 AND MY < 16 THEN
262 FN4     GOSUB GetTile
263 Ga      GOSUB DisplayTiles
264 XW      GOTO LOOP
265 D62   END IF
266 jm    IF MX > 490 AND MX < 554 AND MY > 1 AND MY < 16 THEN
267 Dc4     IF SCE <> -1 THEN PUT (X1(SC),Y1(SC)),Zahl%,PSET
268 d5      GOTO ComputerPlayer
269 HA2   END IF
270 19    IF MX > 1 AND MX < 70 AND MY > 1 AND MY < 16 THEN Patt
271 6U    IF MX > 81 AND MX < 150 AND MY > 1 AND MY < 16 THEN
272 5C4     SAY Tschues$,Laber%
273 94      END
274 MF2   END IF
275 pw    IF MX > 161 AND MX < 250 AND MY > 1 AND MY < 16 THEN
          GOSUB Information
276 EP    IF MY > 15 AND MY < 150 THEN
277 Lh4     FOR t = 0 TO 1
278 VX6       X1(t) = PX(t) + KAX(Richtung(t)) : X2(t) = PX(t) + K
              BX(Richtung(t))
279 rp        Y1(t) = PY(t) + KAY(Richtung(t)) : Y2(t) = PY(t) + K
              BY(Richtung(t))
280 dx        IF MX > X1(t) AND MX < X2(t) AND MY > Y1(t) AND MY
              < Y 2(t) THEN
281 fm8         IF SC <> t AND SC > -1 THEN PUT (X1(SC),Y1(SC)),
                Zahl%,PSET
282 HT          IF SC <> t  THEN
283 R2A           SC = t : Zeichen$ = Wert$(t) :SCE = t
284 rq            GET (X1(t),Y1(t))-(X2(t),Y2(t)),Zahl%
285 dp            LINE(X1(t),Y1(t))-(X2(t),Y2(t)),1,b
286 SX            PAINT(X1(t)+1,Y1(t)+1),2,1
287 ZS8         END IF
288 aT6       END IF
289 St4     NEXT t
290 cV2   END IF
291 Re    IF MX > 35 AND MX < 35*Anzahl(1)+30 AND MY > 150 AND
          MY <200 THEN
292 OP4     ak = INT(MX/35)
293 Vw      SideTaken = -1
294 dY      IF LEFT$(Stapel$(1,ak),1) = Zeichen$ THEN SideTaken = 1
295 A5      IF RIGHT$(Stapel$(1,ak),1) = Zeichen$ THEN SideTaken = 0

296 Z8      IF SideTaken = -1 THEN
297 8j6       SAY Lament$,Laber% : GOTO LOOP
298 kd4     END IF
299 Q92   ELSE
300 764     GOTO LOOP
301 ng2   END IF
302 YN    PUT (X1(SC),Y1(SC)),Zahl%,PSET
303 Z2    dran = 1
304 Uv    SAY OK$ ,Laber% : REM "IHN   OHR D N UHNX."
305 qY0 PutTileOnScreen:
306 3S2   WertRechts = VAL(RIGHT$(Stapel$(dran,ak),1))
307 V9    WertLinks = VAL(LEFT$(Stapel$(dran,ak),1))
308 Ep    IF SideTaken = 1 THEN
309 DV4     Wert$(SC) = RIGHT$(STR$(WertRechts),1)
310 Mp      IF Richtung(SC) = 4 OR Richtung(SC) = 1 THEN SWAP WertLi
            nks, WertRechts
311 cL2   ELSE
312 7g4     Wert$(SC) = RIGHT$(STR$(WertLinks),1)
313 Ps      IF Richtung(SC) = 2 OR Richtung(SC) = 3 THEN SWAP WertLi
            nks, WertRechts
314 Ot2   END IF
315 9n    PX(SC) = PX(SC) + ADX(SC) : PY(SC) = PY(SC) + ADY(SC)
316 iT    ZZ(SC) = ZZ(SC) + 1
317 Rn    IF ZZ(0) = 5 AND SC = 0 THEN
318 524     ZR$(0) = "H" : ADX(0) = 0 : ADY(0) = 32 :Richtung(0)=3
319 gl      PX(0) = PX(0) - 33 : PY(0) = PY(0) + 18
320 Gr2   ELSEIF ZZ(0) = 6 AND SC = 0 THEN
321 6U4     ZR$(0) = "V" : Richtung(0) = 4 : ADX(0) = -63 : ADY(0) =
            0
322 el      PX(0) = PX(0) - 30
323 ia      SWAP WertLinks,WertRechts
324 Vv2   ELSEIF ZZ(0) = 15 AND SC = 0 THEN
325 JJ4     ZR$(0) = "H" : Richtung(0) = 1 : ADX(0) = 0 : ADY(0) = -
            33
326 MW      PY(0) = PY(0) - 15 : PX(0) = PX(0) + 30
327 b22   ELSEIF ZZ(0) = 16 AND SC = 0 THEN
328 kY4     ZR$(0) = "V" : Richtung(0) = 2 : ADX(0) = 63 : ADY(0) =
            0
329 wR      PY(0) = PY(0) + 15
330 ph      SWAP WertLinks,WertRechts
331 e52   ELSEIF ZZ(0) = 25 AND SC = 0 THEN
332 Hf4     ZR$(0) = "V" : Richtung(0) = 4 : ADX(0) = -63 : ADY(0) =
            0
333 Nn      PY(0) = PY(0) +17 : PX(0) = PX(0) - 63
334 t13    SWAP WertLinks,WertRechts
335 h82   ELSEIF ZZ(0) = 34 AND SC = 0 THEN
336 sg4     ZR$(0) = "V" : Richtung(0) = 2 : ADX(0) = 63 : ADY(0) =
            0
337 Vm      PX(0) = PX(0) + 63 : PY(0) = PY(0) - 8
338 xp      SWAP WertLinks,WertRechts
339 cE2   ELSEIF ZZ(1) = 5 AND SC = 1 THEN
340 qk4     ZR$(1) = "H" : Richtung(1) = 1 : ADX(1) = 0 : ADY(1) = -
            33
341 NJ      PX(1) = PX(1) +30 : PY(1) = PY(1) - 15
342 JN2   ELSEIF ZZ(1) = 7 AND SC = 1 THEN
343 I04     ZR$(1) = "V" : Richtung(1) = 2 : ADX(1) = 63 : ADY(1) =
            0
```

```
344 xH      PY(1) = PY(1) + 33 : PX(1) = PX(1) + 33
345 4w      SWAP WertLinks,WertRechts
346 OT2   ELSEIF ZZ(1) = 16 AND SC = 1 THEN
347 7g4     ZR$(1) = "H" : Richtung(1) = 3 : ADX(1) = 0 : ADY(1) = 3
            3
348 1C      PX(1) = PX(1) -33 : PY(1) = PY(1) + 17
349 6a2   ELSEIF ZZ(1) = 17 AND SC = 1 THEN
350 tB4     ZR$(1) = "V" : Richtung(1) = 4 : ADX(1) = -63 : ADY(1) =
            0
351 h6      PX(1) = PX(1) - 63 : PY(1) = PY(1) - 18
352 B3      SWAP WertLinks,WertRechts
353 9d2   ELSEIF ZZ(1) = 26 AND SC = 1 THEN
354 TB4     ZR$(1) = "V" : Richtung(1) = 2 : ADX(1) = 63 : ADY(1) =
            0
355 5N      PY(1) = PY(1) - 18 : PX(1) = PX(1) +63
356 F7      SWAP WertLinks,WertRechts
357 Cg2   ELSEIF ZZ(1) = 35 AND SC = 1 THEN
358 1J4     ZR$(1) = "V" : Richtung(1) = 4 : ADX(1) = -63 : ADY(1) =
            0
359 HN      PX(1) = PX(1) - 63 : PY(1) = PY(1) + 8
360 JB      SWAP WertLinks,WertRechts
361 1e2   END IF
362 3B    CALL Stein (PX(SC), PY(SC), WertLinks, WertRechts, ZR$(SC)
          )
363 kx    AKTT(dran) = ak
364 VO    GOSUB Cleanup
365 dw    Zeichen$ = "O" : SC = -1
366 Nh    IF Anzahl(dran) = 0 THEN Schluss
367 us    IF dran = 1 THEN
368 xH4     GOSUB DisplayTiles
369 Gi      GOTO ComputerPlayer
370 ZI2   ELSE
371 GF4     GOTO LOOP
372 wp2   END IF
373 QHO Patt:
374 uG2   FOR t = 0 TO 1
375 eJ4     Punkte(t) = 0
376 1S      FOR a = 1 TO Anzahl(t)
377 HE6       Punkte(t) = VAL(RIGHT$(Stapel$(t,a),1))+VAL(LEFT$(Stap
              el$(t,a),1))
378 HP4     NEXT a
379 uL2   NEXT t
380 ET    IF Punkte(1) > Punkte(0) THEN dran = 1 : ELSE dran = 0
381 Ps0 GOTO Schluss
382 QK  Schluss:
383 852   IF dran = 0 THEN
384 t24     a$="IH/C  /HAHB EH G EHW OHN EHN   UHN D   D UH   F EHR
            L OHR EHN ."
385 oX2   ELSE
386 cy4     a$ ="D UH  /HAHS T   M IH/C  B EHS IHG T ."
387 B42   END IF
388 AW    SAY a$,Laber%
389 d5    FOR t = 0 TO 7
390 LU4     PALETTE t,0,0,0
391 6X2   NEXT t
392 bn0 GOTO Main
393 8u  Leuchten:
394 dh2   FOR w! = 0 TO 1 STEP .005
395 rd4     PALETTE 2,w!,w!,w!
396 Mz      FOR u = 0 TO 20: NEXT u
397 Lq2   NEXT w!
398 1O    FOR w! = 1 TO 0 STEP -.005
399 vh4     PALETTE 2,w!,w!,w!
400 Q3      FOR u = 0 TO 20: NEXT u
401 Pu2   NEXT w!
402 qS0 RETURN
403 nQ  Vorspann:
404 4A2   CLS
405 Le    COLOR 2,0
406 Te    LOCATE 10,33
407 RB    PRINT "AMIGA DOMINO"
408 1e    GOSUB Leuchten
409 9F    CLS
410 ya0 RETURN
411 uL  Information:
412 RU2   a$ = "-----------Information zum Domino------------"
413 EO    WINDOW 2,a$,(140,20)-(500,170),0,2
414 Rj    COLOR 1,0
415 Oh    PRINT "Das Amiga Domino läuft unter den Regeln des "
416 gy    PRINT "herkömmlichen Dominos, bis darauf, daß es für"
417 jd    PRINT "den Spieler möglich ist auf seinen Zug zu "
418 Mj    PRINT "verzichten (falls er keinen Wert auf Fairplay"
419 Tr    PRINT "legt) und klickt, wie wenn er keinen Stein "
420 GO    PRINT "ablegen könnte das Feld ";
421 rX    COLOR 1,2 : PRINT "WEITER";
422 bQ    COLOR 1,0 : PRINT " an."
423 OZ    PRINT "Um einen Stein regulär abzulegen, klickt man "
424 ZM    PRINT "zuerst das Ende der Schlange an, an das er "
425 IB    PRINT "angelegt wird und dann den eigenen Stein. Muß"
426 Lx    PRINT "ein Stein gezogen werden, ist der Stapel in "
427 gN    PRINT "der rechten oberen Ecke anzuklicken. "
428 kQ    PRINT "Wenn das Spiel einmal nicht aufgeht, oder"
429 IU    PRINT "der Spieler ein neues Spiel beginnen will "
430 g6    PRINT "wird auf ";
431 K1    COLOR 1,2 : PRINT "PATT";
432 Wr    COLOR 1,0 : PRINT " gedrückt, der Computer "
433 Lr    PRINT "berechnet anhand der Augenzahl den Sieger."
434 3p    PRINT "Zum beenden des Programmes dient ";
435 07    COLOR 1,2 : PRINT "ENDE";
436 Jh    COLOR 1,0 : PRINT "."
437 P2    PRINT
438 sB    COLOR 2,0
439 mc    PRINT "           Viel Spaß beim Domino !";
440 QM    CALL Maus(a,b)
441 Jz    WINDOW CLOSE 2
442 U60 RETURN
443 g3  SUB Stein(cx,cy,Erste,Zweite, Modu$) STATIC
444 6n1  SHARED Eins%(),Zwei%(),Drei%(),Vier%(),Fuenf%(),Sechs%(),Mu
         sterA%()
445 3q   SHARED XAR,XAL,YAR,YAL
446 Lx2   PATTERN &HFFFF,MusterA%
447 jy    IF XAR < 0 OR XAL >0 OR YAR < 0 OR YAL > 0 THEN SWAP E
          rste,Zweite
448 Uo    IF Modu$ = "V" THEN
449 H04     LINE(cx,cy)-(cx+60,cy+15),7,bf
450 fD      AREA (cx+61,cy+1)
451 oQ      AREA (cx+63,cy+3)
452 1y      AREA (cx+63,cy+17)
453 sl      AREA (cx+61,cy+15)
454 Kh      COLOR 6,0
455 OS      AREAFILL 0
456 fI      AREA (cx,cy+16)
457 xq      AREA (cx+60,cy+16)
458 74      AREA (cx+63,cy+17)
459 7L      AREA (cx+5,cy+17)
460 Nj      COLOR 5,0
461 UY      AREAFILL 0
462 tG      DeltaX = 30 : DeltaY = 0
463 4n2   ELSE
464 6j4     LINE(cx,cy)-(cx+30,cy+30),7,bf
465 cM      AREA (cx+31,cy+1)
466 1Z      AREA (cx+33,cy+3)
467 nt      AREA (cx+33,cy+32)
468 eg      AREA (cx+31,cy+30)
469 Zw      COLOR 6,0
470 dh      AREAFILL 0
471 dD      AREA (cx,cy+31)
472 j1      AREA (cx+30,cy+31)
473 tz      AREA (cx+33,cy+32)
474 OP      AREA (cx+5,cy+32)
475 cy      COLOR 5,0
476 jn      AREAFILL 0
477 Gr      DeltaX = 0 : DeltaY = 15
478 eX2   END IF
479 Ri    IF Erste = 1 THEN PUT (cx,cy),Eins%,PSET
480 T1    IF Erste = 2 THEN PUT (cx,cy),Zwei%,PSET
481 ai    IF Erste = 3 THEN PUT (cx,cy),Drei%,PSET
482 Wx    IF Erste = 4 THEN PUT (cx,cy),Vier%,PSET
483 oE    IF Erste = 5 THEN PUT (cx,cy),Fuenf%,PSET
484 Aa    IF Erste = 6 THEN PUT (cx,cy),Sechs%,PSET
485 UM    IF Zweite = 1 THEN PUT (cx+DeltaX,cy+DeltaY),Eins%,PSET
486 Xg    IF Zweite = 2 THEN PUT (cx+DeltaX,cy+DeltaY),Zwei%,PSET
487 fO    IF Zweite = 3 THEN PUT (cx+DeltaX,cy+DeltaY),Drei%,PSET
488 ce    IF Zweite = 4 THEN PUT (cx+DeltaX,cy+DeltaY),Vier%,PSET
489 vw    IF Zweite = 5 THEN PUT (cx+DeltaX,cy+DeltaY),Fuenf%,PSET
490 IJ    IF Zweite = 6 THEN PUT (cx+DeltaX,cy+DeltaY),Sechs%,PSET
491 BV    IF Modu$ = "V" THEN
492 sr4     LINE (cx+29,cy)-(cx+30,cy+15),4,bf
493 YH2   ELSE
494 2z4     LINE (cx,cy+15)-(cx+30,cy+15),4,bf
495 vo2   END IF
496 240 END SUB
497 dA  SUB Maus(MX,MY) STATIC
498 Ff2   WHILE MOUSE(0) = 0
499 UI    WEND
500 M9    MX = MOUSE(3) : MY = MOUSE(4)
501 yr    WHILE MOUSE(0) <> 0
502 XL    WEND
503 9B0 END SUB
(C) 1987 M&T
```

Listing. »Domino«: Bitte mit dem Checksummer (Seite 66) eingeben

C-Kurs für Anfänger



Die Programmiersprache C stellt uns mit den zusammengesetzten Datentypen ein mächtiges Werkzeug zur Verfügung. In diesem Kursteil werden außer diesen Datentypen noch die sehr wichtigen Operatoren unter C gründlich beleuchtet.

Nachdem wir in der letzten Ausgabe unseres C-Kurses im AMIGA-Magazin die einfachen Datentypen kennengelernt haben, befassen wir uns in diesem Kursteil zuerst mit den zusammengesetzten Datentypen. Wenn Sie die einfachen oder, wie sie auch genannt werden, skalaren Typen verstanden haben, wird Ihnen das Gebiet der zusammengesetzten Datentypen keine großen Schwierigkeiten mehr bereiten. Sie setzen sich grundsätzlich nur aus uns bereits bekannten Datentypen zusammen.

Das Array (Feld) besteht aus mehreren gleichartigen Elementen. Das heißt in einem Array kann zum Beispiel eine Gruppe von Integervariablen zusammengefaßt werden. Die folgenden Datentypen dürfen in Arrays zusammenfaßt werden:

- weitere Arrays oder
- alle einfachen Datentypen oder
- Zeiger oder
- Strukturen oder
- Unionen

Strukturen und Unionen sind ebenfalls zusammengesetzte Datentypen. Diese werden später genauer besprochen. Wir befassen uns zunächst mit den eindimensionalen Arrays oder Vektoren (siehe dazu auch Bild 1).

Stellen Sie sich einfach eine Reihe von Schubladen vor. Diese Schubladenreihe soll ein Array darstellen. In jeder dieser Schubladen befindet sich ein Element dieses Arrays. Wir müssen dem Compiler genau mitteilen, wie viele Schubladen wir benötigen und was wir darin zu deponieren wünschen; zum Beispiel eine Gruppe von int-Variablen.

Die Definition eines Arrays, das aus Integerwerten besteht, sieht folgendermaßen aus:

```
int name[ELEMENTE];
```

Wobei »int« der Datentyp der Arrayelemente, »name« der Name des Arrays und »ELEMENTE« die Anzahl der Arrayelemente (Schubladen) darstellt. In eckigen Klammern, den sogenannten Indexoperatoren, wird angegeben, wie viele Elemente das Array aufnehmen soll. Die Anzahl der Elemente muß durch einen konstanten Ausdruck oder eine symbolische Konstante repräsentiert werden. In Basic würde das so aussehen:

```
DIM name(ELEMENTE)
```

Ausdrücke sind grob gesagt Operationen mit skalaren Objekten, beispielsweise die Division zweier Zahlen. Aber das soll uns heute nicht weiter belasten, denn dieses Thema ist Stoff des nächsten Kursteils. Wichtig ist, daß konstante Ausdrücke konstante Werte liefern.

Erlaubt sind folgende Arraydefinitionen:

```
100
110 #define ELEMEN
    TE 100
120
130 int zahl;
140
150 int name_a[10];
160 int name_b[ELEMEN
    TE];
170 int name_c[ELEMENTE
    +10];
180
190 int name_e[]= {0,1,
    2};
200
```

In Zeile 150 bis 170 werden Arrays mit jeweils 10, 100 und 110 Elementen initialisiert. Die in eckigen Klammern dargestellte Länge des Arrays ist immer ein konstanter Wert.

Falsch hingegen wäre diese Konstruktion:

```
210 int name_d[zahl];
    /* falsch! */
```

»zahl« ist eine Integervariable, die in Zeile 130 definiert wurde. Die Anzahl der Elemente eines Arrays ist jedoch nicht variabel und muß somit durch einen konstanten Wert dargestellt werden!

In Zeile 190 sehen Sie eine Definition, bei der die Anzahl der Elemente automatisch ermittelt wird. Eine solche Initialisierung ist nur bei globalen Arrays erlaubt. Das heißt Zeile 190 muß außerhalb jeder Funktion stehen, oder es muß die Speicherklasse »static« mit angegeben werden:

```
190 static int name_e[]=
    {0,1,2};
```

Die gleiche Wirkung erzielen Sie innerhalb eines Funktionsblocks mit den folgenden Zeilen:

```
200 {
210 int name_e[3];
220
230 name_e[0]=0;
240 name_e[1]=1;
250 name_e[2]=2;
260 }
```

Beachten Sie, daß die Elemente immer mit dem Index 0 beginnen. Daraus folgt, daß der höchste Index immer um eins kleiner ist, als die Zahl der Elemente:

```
float dez_bruch[10];
```

Dieses float-Array kann zehn float-Zahlen (Elemente) aufnehmen. Es sind dies die Zahlen, die durch die Namen dez_bruch[0] bis dez_bruch[9] dargestellt werden.

Achten Sie hierauf in Ihrer Programmierpraxis, denn wie verführerisch ist es doch, hier die (nicht existierende) Variable dez_bruch[10] anzusprechen.

Eine Sonderform des Arrays sind die Zeichenarrays (Zeichenketten/Strings). Sie können diese auf zwei Arten initialisieren:

```
100
110 char string_a[6]={'
    A','m','i','g','a','\0'};
120 char string_b[6]="
    Amiga";
130
```

Beide Vereinbarungen haben die gleiche Wirkung. Aber Vorsicht: Im ersten Fall muß das Nullbyte angehängt werden! Im zweiten Fall geschieht dies zwar automatisch, aber es muß trotzdem ein Element mehr angegeben werden — in der letzten Schublade eines Zeichenarrays finden Sie immer ein Nullbyte! Hier noch die globale Initialisierung einer Zeichenkette:

```
140 static char string_c
    []="Amiga";
```

Es wurde ein sechs Zeichen langes Array mit dem Inhalt

```
string_c[0]='A';
string_c[1]='m';
string_c[2]='i';
string_c[3]='g';
string_c[4]='a';
string_c[5]='\0';
```

angelegt.

Zeichenketten können aber auch über Zeiger angesprochen werden. Dies sei an dieser Stelle erwähnt, um die »Ver-

Teil 3

KURSÜBERSICHT

Unser C-Kurs soll allen Lesern einen Einstieg in die Programmiersprache C bieten. Kenntnisse in anderen Sprachen sind nützlich, aber nicht notwendig. In den einzelnen Kursteilen finden Sie folgende Themen:

TEIL 1: Geschichte von C; das Modulkonzept; Beispielprogramme mit der Funktion »printf«, Syntaxdiagramme

TEIL 2: Einfache Datentypen, aufgeteilt nach Konstanten und Variablen

TEIL 3: Zusammengesetzte Datentypen; Zeiger; Typ-Konvertierung und Operatoren

TEIL 4: Ausdrücke und Anweisungen mit erklärenden Beispielen

TEIL 5: Aufbau von Funktionen; Parameterübergabe und Aufruf mit Programmen

TEIL 6: Der C-Präprozessor und die Bibliotheksfunktionen mit Beispielen und Übungen

wandtschaft« zwischen Zeiger und Arrays darzustellen.

Wie Sie wissen, ist nach der Definition »char *zeiger;« die Variable »zeiger« ein Zeiger auf ein Zeichen. Nach der Zuweisung »zeiger= "String";« enthält »zeiger« nicht etwa die Zeichenkette »String«, sondern einen Zeiger auf das erste Zeichen der Zeichenkette. Der Aufruf »printf("%c",*zeiger);« bewirkt also, daß das Zeichen »S« auf dem Bildschirm ausgegeben wird. Mittels »printf ("%s", zeiger);« jedoch wird die Zeichenkette »String« ausgegeben!

Die einzelnen Zeichen eines Strings sind im Speicher hintereinander abgelegt. Nachdem die Adresse des ersten Zeichens bekannt ist, müßte man doch ohne weiteres auch auf die anderen Zeichen zugreifen können? Dies ist tatsächlich möglich! Zum Beispiel »printf("%c",*(zeiger+1));« gibt das Zeichen »t« aus! Folglich ergibt »printf("%s", zeiger+1);« die Ausgabe »tring«.

Arrays in Arrays

Weiter oben wurde beschrieben, daß Arrays unter anderem auch weitere Arrays enthalten dürfen. Es entsteht dann ein mehrdimensionales Array oder eine Matrix. In jeder unserer Schubladen befindet sich jetzt wieder eine Reihe weiterer Schubladen.

```
int name[10][10];
```

Es wurden hier 10 Arrays definiert, und zwar die von [0] bis [9]. Jedes dieser Arrays beinhaltet 10 Elemente, eben die Elemente [0] bis [9]. Wir haben also Platz für 100 int-Variablen geschaffen. Bei der globalen Definition von mehrdimensionalen Arrays wird die Anzahl der Arrays automatisch ermittelt. Die erste Klammer darf also leer bleiben. Als Beispiel die statische Initialisierung einer Matrix:

```
110 static int name[][3]
    ={
120       { 1,2,3 },
130       { 4,5,6 }
140 };
```

oder

```
120 static int name[][3]
    = {1,2,3,4,5,6};
```

Wegen der Übersichtlichkeit und der besseren Lesbarkeit wird man die erste Art der statischen Initialisierung eines mehrdimensionalen Arrays vorziehen. Es wurden in jedem Fall zwei Arrays mit je drei Elementen initialisiert. Hier die explizite Wertzuweisung der einzelnen Elemente:

```
210 name[0][0]=1;
220 name[0][1]=2;
230 name[0][2]=3;
240
250 name[1][0]=4;
260 name[1][1]=5;
270 name[1][2]=6;
```

Sie sehen, obwohl ein Array eine zusammengehörende Einheit bildet, können sehr leicht einzelne Elemente darin verändert werden.

Angenommen, Sie schreiben ein Programm, das die Adressen einiger Ihnen bekannter Personen verwalten soll. Sie definieren ein mehrdimensionales Array in der folgenden Art:

```
char adresse[3][13];
```

Drei Arrays, jeweils eines für Name, Straße und Ort, mit je 19 Elementen wurden vereinbart. Egal wie viele Zeichen Sie belegen, es werden pro Array immer 12 Zeichen und damit auch ebenso viele Speicherstellen reserviert:

```
adresse[0]: Kurt B.______
adresse[1]: Am Berg______
adresse[2]: 8000 München
```

Der Unterstrich (_) stellt zur Verdeutlichung ein unbenutztes Element eines Arrays dar. Sie sehen, diese Art Zeichenketten zu verwalten, kostet unnötigen Speicherplatz. Abhilfe schafft hier das Zeiger-Array. Die Definition:

```
char *adresse[3];
```

Mit dieser Lösung erhalten Sie drei Zeiger auf Zeichenketten, die unterschiedlich lang sein dürfen:

```
adresse[0]="Kurt B.";
adresse[1]="Am Berg";
adresse[2]=
"8000 München";
```

Es wird nur soviel Speicher reserviert, wie tatsächlich notwendig ist. Der Nachteil, den Sie hierbei in Kauf nehmen müssen ist, daß Zeiger-Arrays nicht statisch initialisiert werden können.

Die Struktur vereint verschiedene Datentypen

Strukturen sind für einen C-Programmierer, der sich mit dem Amiga beschäftigen möchte, sehr wichtig. Bei diesem Computer läuft einfach alles (und alles einfach) über Strukturen. Deshalb soll dieses Kapitel auch ausführlich diskutiert werden. Im Gegensatz zu den Arrays können in einer Struktur verschiedene Datentypen vereint werden. Erlaubt sind alle Typen, die auch in Arrays erlaubt sind und zusätzlich noch die sogenannten Bitfelder (siehe unten).

Die Definition einer Struktur ist der der Aufzählungstypen nicht unähnlich:

```
100
110 struct datum{
120       int tag;
130       char *monat;
140       int jahr;
150 }heute;
160
```

In Zeile 110 wird die Definition einer Struktur mit dem Schlüsselwort »struct« (von structure) eingeleitet. Es folgt der Strukturtypname — im Beispiel datum — dahinter die öffnende geschweifte Klammer.

In Zeile 120 bis 140 werden die einzelnen Komponenten der Struktur »datum« aufgezählt.

Die Zeile 150 enthält neben der schließenden geschweiften Klammer einen Variablennamen des Typs »datum«. Bild 2 verdeutlicht noch einmal die Syntax einer Struktur. Wie auch bei den Aufzählungstypen ist die Angabe eines oder auch mehrerer Variablennamen erlaubt, aber nicht zwingend notwendig. Nach der Definition des Strukturtyps könnten zum Beispiel mit

```
170 struct datum morgen;
```

oder mit

```
180 struct datum *ge
    stern;
```

weitere Variablen oder Zeiger des gleichen Strukturtyps vereinbart werden. Dies ist jedoch nur dann möglich, wenn bei der Stukturdefinition ein Strukturtypname, im Beispiel »datum«, angegeben wurde.

Wie beim Array ist eine statische Initialisierung möglich. Hier die vier Möglichkeiten anhand eines anderen Beispiels:

1) außerhalb einer Funktion und direkt bei der Definition der Struktur

```
110 struct fehlermeldun
    gen{
120       int code;
130       char *meldung;
140 }dos[]={
150       103,"Speicher
    platz nicht ausrei
    chend",
160       121,"Datei ist
    kein Objektmodul",
370       226,"Diskette
    nicht eingelegt"
380 }
```

2) außerhalb einer Funktion und bei der Vereinbarung weiterer Strukturvariablen

```
100
110 struct fehlermeldun
    gen dos[]={
120       103,"Speicher
    platz nicht ausrei
    chend",
130       121,"Datei ist
    kein Objektmodul",
...       ...
140       226,"Diskette
    nicht eingelegt"
150 }
```

3) und 4) innerhalb einer Funktion wie unter den beiden ersten Punkten, nur mit dem Verweis auf die Speicherklasse static. Es wurde ein Strukturarray mit Fehlercode und Fehlermeldung initialisiert. Der Zugriff auf die einzelnen Komponenten einer Struktur ist möglich mit

```
error=dos[0].code;
```

Den Punkt zwischen »dos[0]« und »code« nennt man Auswahloperator. Mit seiner Hilfe können Sie einzelne Werte in einer Struktur verändern oder lesen. »dos[0]« ist der Name einer Struktur und »code« eine Komponente davon. Der Auswahloperator ».« ist gültig für Strukturnamen.

Da aber Zeiger auf Strukturen relativ häufig vorkommen, hat man hierfür einen eigenen Auswahloperator eingeführt. Es ist dies sinnigerweise ein Pfeil, bestehend aus dem Minuszeichen und dem Größerzeichen (->). Die Zeile kann also ersetzt werden durch:

```
450 zeiger->code=205;
```

Wobei Zeiger ein Pointer auf »dos[0]« ist. Weiter oben wurde erwähnt, daß Komponenten von Strukturen wieder Strukturen sein können. Hier ein Beispiel, das in einer Adressenverwaltung Verwendung finden könnte:

```
110 struct adresse{
120       char *name;
130       char *straße;
140       char *ort;
150       struct zweite_
          adresse z_adr;
160 }person;
```

Die Komponenten »name«, »straße« und »ort« sind Zeiger auf entsprechende Zeichenketten. Die Struktur »zweite_adresse« könnte beispielsweise Ort und Straße der Zweit-

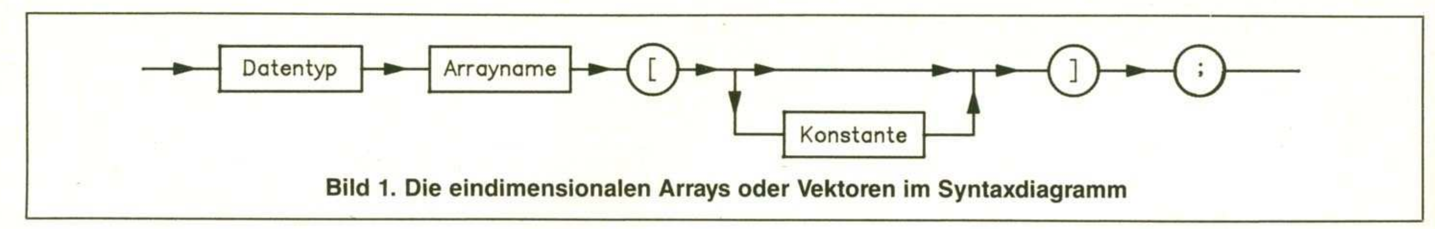


Bild 1. Die eindimensionalen Arrays oder Vektoren im Syntaxdiagramm

wohnung einer Person enthalten. »z_adr« ist eine Variable dieser Struktur. Auf die Daten dieser zweiten Struktur können Sie wie bereits gewohnt zugreifen:

```
200 person.z_adr.ort="
    8000 München";
```

Strukturen können sogar Zeiger auf sich selbst beinhalten. Man spricht dann von einer rekursiven Struktur. An dieser Stelle soll ein Beispielprogramm das bisher Gelernte vertiefen. Das Programm »Pref« (Listing 1) gibt einige der zuletzt eingegebenen Werte der Preferences-Struktur auf dem Bildschirm aus. Diese Struktur enthält die Werte, die Sie mit dem Programm Preferences eingestellt haben.

Dieses kleine Programm wollen wir nun besprechen. Die Präprozessoranweisung »#include ...« in Zeile 150 bewirkt, daß Zeile 150 durch das File »intuition.h« aus dem Unterverzeichnis »intuition« ersetzt wird. Hier finden Sie auch die erwähnte Struktur Preferences wieder.

In Zeile 170 und 180 wird jeweils ein Zeiger beziehungsweise eine Stukturvariable vereinbart. Es folgen die Initialisierung mehrerer Zeiger-Arrays.

In Zeile 440 wird eine Library geöffnet. Wenn dies nicht gelingt, wird das Programm beendet. Solche Konstruktionen besprechen wir zu einem späteren Zeitpunkt ausführlich. Betrachten Sie sie hier lediglich als eine notwendige Vorbereitung.

Die Library wird in der Funktion »ende()« (Zeile 540) wieder geschlossen. Mit dem Aufruf der Funktion »exit()« wird das Programm beendet.

Es folgt nun die Hauptfunktion von »Pref«. Eine Integer-Variable der Speicherklasse auto wird definiert. Nach dem Aufruf der Funktion »start()« wird die Intuition-Funktion »GetPrefs()« aufgerufen. Wie auch bei »printf()« müssen wir hier verschiedene Parameter übergeben.

Ab Zeile 730 werden mit Hilfe des Auswahloperators ».« die eben gelesenen Werte auf dem Bildschirm ausgegeben. Zeile 750 zeigt den Zugriff auf eine Struktur innerhalb der Preferences-Strukur.

Ansonsten dürfte nur noch die bedingte Bewertung in Zeile 860 Fragen aufwerfen — aber nicht mehr lange, denn diese wird noch in diesem Kursteil behandelt.

Das Bitfeld findet immer da Einsatz, wo es gilt, verschiedene Daten auf engstem Raum unterzubringen. Auch bei einem Amiga, und besonders in der Grundversion mit 512 KByte, können bei größeren Programmen und einem etwas lässigen Programmierstil Speicherplatzprobleme auftreten.

Struktur im kleinen — das Bitfeld

Wie Sie vielleicht schon wissen, besteht ein Kilobyte aus 1024 Byte. Jedes Byte wiederum setzt sich aus 8 einzelnen Bits zusammen.

Um zum Beispiel einen Ein-/Ausschalter für eine bestimmte Funktion zu programmieren, wird man eine »short int«-Variable als Flag definieren. Aber eigentlich würde doch auch ein einziges Bit, welches ja die Zustände »an« oder »aus« darstellen kann, für diese Aufgabe ausreichen.

Weiter unten werden wir sehen, daß C über eine Menge von Operatoren verfügt, die es ermöglichen, auf Bitebene zu programmieren. Darüber hinaus besteht aber noch die Möglichkeit, sogenannte Bitfelder zu definieren:

```
110 struct{
120       unsigned FLAG:1;
130       unsigned : 5;
140       unsigned SCHAL-
          TER: 2;
150 }name={0,0,0};
```

Es handelt sich hier um die statische Initialisierung einer Bitfeldgruppe namens »name«. Die Definition ist, bis auf den Doppelpunkt nach jedem Namen eines Bitfeldes, einer Strukturdefinition gleich.

Das Bitfeld »Flag« ist, darüber gibt die Zahl hinter dem Doppelpunkt Auskunft, ein Bit breit und kann somit die Werte 0 oder 1 enthalten.

Nach dieser Vereinbarung folgt eine Zeile, in der kein Bitfeldname angegeben wurde. Dieses Bitfeld kann nicht mehr angesprochen werden, die fünf reservierten Bits dienen lediglich noch als Platzhalter. Das letzte Bitfeld der Gruppe »name« wird mit »Schalter« angesprochen. »Schalter« kann, da zwei Bit breit, Werte von 0 bis 3 (Tabelle 1) annehmen. In Zeile 150 werden alle Bitfelder der Gruppe »name« mit Null initialisiert.

Bitfelder können nur in vorzeichenlosen Integerobjekten gebildet werden:

```
unsigned short FLAG: 1;
unsigned FLAG: 1;
unsigned long FLAG: 1;
```

Man kann Bitfelder wie Strukturkomponenten ansprechen:

```
name.FLAG=1;
```

Die Adresse eines Bitfeldes kann nicht mit dem Adreßoperator (&) festgestellt werden. Eine weitere Möglichkeit, Daten dicht gepackt zu verwalten, bietet C mit der Union:

```
110 union typ{
120       int komponente1;
130       float komponen
          te2;
140       char komponen
          te3;
150 }name;
```

Auch hier steht wieder, wenn man von dem Schlüsselwort »union« absieht, die gleiche Definition wie bei der Struktur. Die Komponenten einer Union können vom gleichen Datentyp sein wie die einer Struktur.

Eigene Namen

Auch werden sie über die uns bekannten Auswahloperatoren angesprochen. Der einzige Unterschied ist, daß alle Daten innerhalb einer Union auf dem gleichen Speicherplatz untergebracht sind.

Daraus folgt natürlich, daß nur immer die Komponente angesprochen werden kann, die zuletzt in den Speicher hineingeschrieben wurde. Der Vorteil dieser Verwaltung liegt klar auf der Hand: es wird nur soviel Speicher reserviert, wie die größte Komponente innerhalb einer Union benötigt.

Um die Lesbarkeit von Programmen zu verbessern, erlaubt C die Einführung eigener Typennamen.

```
typedef char ZEICHEN;
```

Nach dieser Zeile ist Definition »ZEICHEN a;« erlaubt. Sie ist gleichbedeutend mit »char a;«! Ein anderes Beispiel:

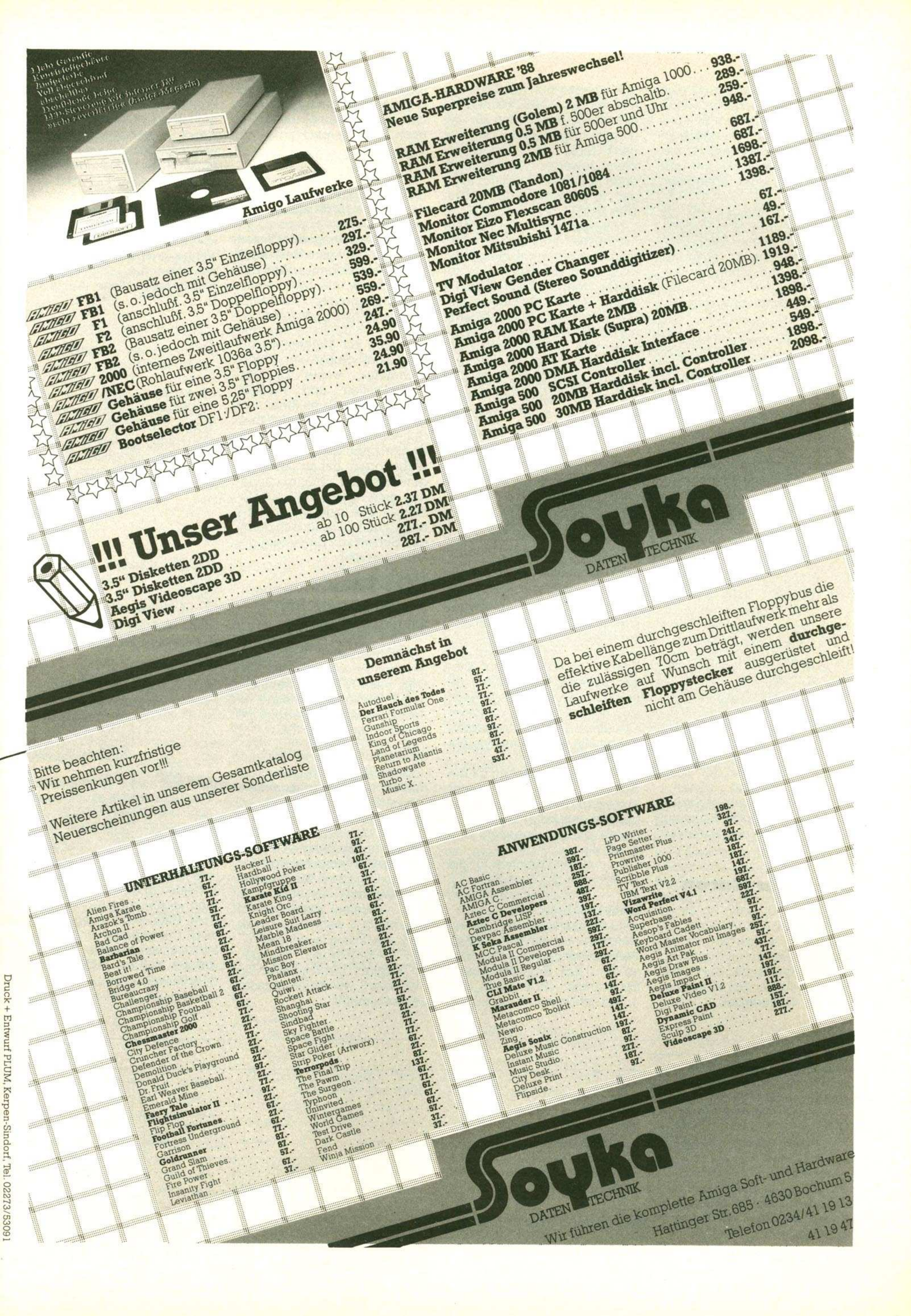
```
110 typedef char
      FILENAME[31];
120
130 FILENAME fname;
```

In Zeile 130 wurde ein Zeichenarray, das 30 Zeichen (und das Nullbyte) aufnehmen kann, definiert. Schauen Sie sich doch einmal das Headerfile (Files, die mit Hilfe der »#include« Anweisung in ein C-Quellprogramm eingebunden werden.) »exec/types.h« an. Hier werden eine ganze Reihe neuer Datentypnamen für den Amigaprogrammierer festgelegt. Eine Gegenüberstellung der speziellen Amiga-Typnamen und der C-Typnamen zeigt Tabelle 2.

Manchmal kommt es vor, daß zum Beispiel die Werte einer short int-Variablen und einer long int-Variablen addiert werden müssen. In einem solchen Fall findet eine automatische Typumwandlung statt. Diese Typumwandlung bezieht sich jedoch nur auf Werte der Variablen während dieser einen Operation.

Die short int-Variable in unserem Beispiel ist danach immer noch vom Typ short int; auch die long int-Variable behält Ihren Datentyp bei. Allgemein kann man folgendes festhalten: Bei arithmetischen Operationen mit Datentypen unterschiedlicher Länge wird der Typ mit dem geringeren Speicherbedarf in den Typ mit dem höheren Speicherbedarf umgewandelt. Das Ergebnis dieser Operation ist ebenfalls von diesem Typ. Außer dieser Regel ist noch zu beachten, daß char und short immer zu int und float immer zu double konvertiert wird. Bei Funktionsaufrufen werden wie gewohnt weiterhin Arraynamen zu Zeigern gewandelt.

Neben dieser automatischen Typkonvertierung besteht die Möglichkeit, mit Hilfe



Amigo Laufwerke
AMIGO FB1 (Bausatz einer 3.5" Einzelfloppy) 275.-
AMIGO FB1 (s. o. jedoch mit Gehäuse) 297.-
AMIGO F1 (anschlußf. 3.5" Einzelfloppy) 329.-
AMIGO F2 (anschlußf. 3.5" Doppelfloppy) 599.-
AMIGO FB2 (Bausatz einer 3.5" Doppelfloppy) 539.-
AMIGO FB2 (s. o. jedoch mit Gehäuse) 559.-
AMIGO 2000 (internes Zweitlaufwerk Amiga 2000) 269.-
AMIGO /NEC (Rohlaufwerk 1036a 3.5") 247.-
AMIGO Gehäuse für eine 3.5" Floppy 24.90
AMIGO Gehäuse für zwei 3.5" Floppies 35.90
AMIGO Gehäuse für eine 5,25" Floppy 24.90
AMIGO Bootselector DF1:/DF2: 21.90
AMIGA-HARDWARE '88
Neue Superpreise zum Jahreswechsel!
RAM Erweiterung (Golem) 2 MB für Amiga 1000 938.-
RAM Erweiterung 0.5 MB f. 500er abschaltb. 289.-
RAM Erweiterung 0.5 MB für 500er und Uhr 259.-
RAM Erweiterung 2MB für Amiga 500 948.-
Filecard 20MB (Tandon) 687.-
Monitor Commodore 1081/1084 687.-
Monitor Eizo Flexscan 8060S 1698.-
Monitor Nec Multisync 1387.-
Monitor Mitsubishi 1471a 1398.-
TV Modulator 67.-
Digi View Gender Changer 49.-
Perfect Sound (Stereo Sounddigitizer) 167.-
Amiga 2000 PC Karte 1189.-
Amiga 2000 PC Karte + Harddisk (Filecard 20MB) 1919.-
Amiga 2000 RAM Karte 2MB 948.-
Amiga 2000 Hard Disk (Supra) 20MB 1398.-
Amiga 2000 AT Karte 1898.-
Amiga 2000 DMA Harddisk Interface 449.-
Amiga 500 SCSI Controller 549.-
Amiga 500 20MB Harddisk incl. Controller 1898.-
Amiga 500 30MB Harddisk incl. Controller 2098.-
!!! Unser Angebot !!!
3.5" Disketten 2DD ab 10 Stück 2.37 DM
3.5" Disketten 2DD ab 100 Stück 2.27 DM
Aegis Videoscape 3D 277.- DM
Digi View 287.- DM
Soyka DATEN TECHNIK
Demnächst in unserem Angebot
Autoduel
Der Hauch des Todes
Ferrari Formular One
Gunship
Indoor Sports
King of Chicago
Land of Legends
Planetarium
Return to Atlantis
Shadowgate
Turbo
Music X
87.- 57.- 77.- 77.- 97.- 87.- 87.- 97.- 87.- 77.- 47.- 537.-
Da bei einem durchgeschleiften Floppybus die effektive Kabellänge zum Drittlaufwerk mehr als die zulässigen 70cm beträgt, werden unsere Laufwerke auf Wunsch mit einem durchgeschleiften Floppystecker ausgerüstet und nicht am Gehäuse durchgeschleift!
Bitte beachten: Wir nehmen kurzfristige Preissenkungen vor!!!
Weitere Artikel in unserem Gesamtkatalog
Neuerscheinungen aus unserer Sonderliste
UNTERHALTUNGS-SOFTWARE
Alien Fires
Amiga Karate
Arazok's Tomb
Archon II
Bad Cad
Balance of Power
Barbarian
Bard's Tale
Beat it!
Borrowed Time
Bridge 4.0
Bureaucrazy
Challenger
Championship Baseball
Championship Basketball 2
Championship Football
Championship Golf
Chessmaster 2000
City Defence
Cruncher Factory
Defender of the Crown
Demolition
Donald Duck's Playground
Dr. Fruit
Earl Weaver Baseball
Emerald Mine
Faery Tale
Flightsimulator II
Flip Flop
Football Fortunes
Fortress Underground
Garrison
Goldrunner
Grand Slam
Guild of Thieves
Fire Power
Insanity Fight
Leviathan
77.- 67.- 77.- 77.- 57.- 77.- 67.- 87.- 27.- 67.- 57.- 87.- 27.- 67.- 67.- 67.- 67.- 77.- 27.- 27.- 77.- 27.- 57.- 27.- 97.- 27.- 77.- 97.- 27.- 67.- 27.- 67.- 77.- 87.- 87.- 57.- 67.- 37.-
Hacker II
Hardball
Hollywood Poker
Kampfgruppe
Karate Kid II
Karate King
Knight Orc
Leader Board
Leisure Suit Larry
Marble Madness
Mean 18
Mindbreaker
Mission Elevator
Pac Boy
Phalanx
Quintett
Quiwi
Rockett Attack
Shanghai
Shooting Star
Sindbad
Sky Fighter
Space Battle
Space Fight
Star Glider
Strip Poker (Artworx)
Terrorpods
The Final Trip
The Pawm
The Surgeon
Typhoon
Uninvited
Wintergames
World Games
Test Drive
Dark Castle
Fend
Winja Mission
77.- 97.- 47.- 107.- 67.- 37.- 37.- 77.- 67.- 87.- 67.- 87.- 27.- 57.- 27.- 27.- 87.- 57.- 27.- 77.- 27.- 77.- 57.- 27.- 27.- 77.- 77.- 67.- 27.- 87.- 137.- 67.- 77.- 67.- 67.- 67.- 57.- 37.- 37.-
ANWENDUNGS-SOFTWARE
AC Basic
AC Fortran
AMIGA Assembler
AMIGA C
Aztec C Commercial
Aztec C Developers
Cambridge LISP
Devpac Assembler
K Seka Assembler
MCC Pascal
Modula II Commercial
Modula II Developers
Modula II Regular
True Basic
CLI Mate V1.2
Grabbit
Marauder II
Metacomco Shell
Metacomco Toolkit
Newio
Zing
Aegis Sonix
Deluxe Music Construction
Instant Music
Music Studio
City Desk
Deluxe Print
Flipside
387.- 597.- 187.- 257.- 888.- 487.- 397.- 197.- 137.- 227.- 597.- 297.- 177.- 297.- 67.- 67.- 67.- 147.- 97.- 497.- 147.- 147.- 197.- 87.- 97.- 277.- 187.- 97.-
LPD Writer
Page Setter
Printmaster Plus
Prowrite
Publisher 1000
Scribble Plus
TV Text
UBM Text V2.2
Vizawrite
Word Perfect V4.1
Acquisition
Superbase
Aesop's Fables
Keyboard Cadett
Word Master Vocabulary
Aegis Animator mit Images
Aegis Art Pak
Aegis Draw Plus
Aegis Images
Aegis Impact
Deluxe Paint II
Deluxe Video V1.2
Digi Paint
Dynamic CAD
Express Paint
Sculp 3D
Videoscape 3D
198.- 327.- 97.- 247.- 347.- 187.- 187.- 147.- 197.- 687.- 597.- 227.- 97.- 77.- 97.- 257.- 57.- 437.- 77.- 147.- 197.- 197.- 117.- 888.- 157.- 187.- 277.-
Soyka DATEN TECHNIK
Wir führen die komplette Amiga Soft- und Hardware
Hattinger Str. 685 · 4630 Bochum 5
Telefon 0234/41 19 13
41 19 47
Druck + Entwurf PLUM, Kerpen-Sindorf, Tel. 02273/53091

des cast-Operators »()«, eine Konvertierung explizit vorzunehmen. Dies ist zum Beispiel dann nötig, wenn Sie genau errechnete float-Werte einer Betriebssystemroutine, die nur int-Werte erwartet, übergeben wollen.

Außerdem steigert es die Lesbarkeit Ihrer Programme, wenn Sie mittels des cast-Operators einer automatischen Typkonvertierung vorgreifen.

Hier ein einfaches Beispiel:

```
110 /*
120     typedef - float
     in int und umgekehrt
     konvertieren
130 */
140
150 void main()
160 {
170   int ganze_zahl=100;
180   float dez_bruch
      =11.11;
190   float ergebnis;
200
210   ergebnis = dez
      _bruch + (float)
      ganze_zahl;
220   printf("%f + %d =
      %f\n",dez_bruch,
      ganze_zahl,
      ergebnis);
230   printf("%f * 2 = %d
      gerundet\n",ergeb
      nis,(int)ergebnis
      *2);
240 }
```

In Zeile 210 wird zur float-Zahl »dez_bruch« der float-Wert der int-Zahl »ganze_zahl« addiert. Beachten Sie bitte, daß sich weder Datentyp noch der Wert der Variablen geändert haben.

Die Operatoren

Wird ein Datentyp in einen Datentyp mit geringerem Speicherbedarf konvertiert, so geschieht dies durch Rundung oder Unterdrückung überzähliger Bits, in der Regel auf Kosten der Genauigkeit.

Sie haben nun alle Datentypen der Sprache C kennengelernt. Aber was können Sie mit diesen Kenntnissen anfangen? Einer Variablen einen Wert zuweisen und diesen dann auf dem Bildschirm ausgeben — das kann doch nicht alles sein?

Natürlich nicht! Sie können mit diesen Variablen rechnen, man kann sie vergleichen, verknüpfen, manipulieren und so weiter. Hierfür stellt C eine wahre Vielfalt an Operatoren zur Verfügung. Verschiedene haben Sie ja bereits kennengelernt. Wir unterscheiden zwischen:

- Auswahloperatoren
- arithmetische Operatoren
- Vergleichsoperatoren
- Operatoren für logische Verknüpfungen
- Operatoren für Bitmanipulationen
- Zuweisungsoperatoren
- Inkrement- und Dekrementoperatoren
- andere Operatoren

Begeben wir uns also gleich ins Reich der arithmetischen Operatoren. C bietet insgesamt sechs davon an:

»-« — ein Minuszeichen als Vorzeichen für eine Zahl. Dieser Operator benötigt nur einen Operanden. Es handelt sich somit um einen »unitären Operator«. Unitäre Operatoren haben alle die gleiche Priorität (Tabelle 3). Beispiel: -x.

»-« — ein Minuszeichen für die Subtraktion. Da der Operator in diesem Fall zwei Operanden benötigt, spricht man hier von einem »binären Operator«. Beispiel: x-y.

»+« — ein Pluszeichen für die Addition (binär). Beispiel: x+y.

»*« — Multiplikationszeichen (binär). Beispiel: x*y.

»/« — Divisionszeichen (binär): Beispiel: x/y. Aber Achtung: Werden Integerwerte dividiert, so wird das Ergebnis immer abgerundet. 49/10 ergibt also 4. Soll richtig gerundet werden, müssen Sie mit float-Werten rechnen und dem Ergebnis vor der Umwandlung nach int noch 0,5 addieren:

```
49.0/10.0 ergibt 4.9
4.9+.5 ergibt 5.4
```

Das Ergebnis entspricht dem int-Wert 5, da ja immer abgerundet wird!

»%« — Modulooperator, liefert den ganzzahligen Rest nach einer Division (binär). Zwei Beispiele dafür:

```
49/10 ergibt 9
50/10 ergibt 0
```

Analog zur Mathematik gilt die Regel »Punkt- vor Strichrechnung«, wobei Division, Multiplikation und Modulo, auch wenn man es ihnen wegen der amerikanischen Schreibweise nicht ansieht, Punktrechnungen sind. Punktrechnungen haben also gegenüber den Strichrechnungen eine höhere Priorität (Tabelle 3). Wie in der Mathematik auch, kann diese jedoch durch das Setzen von Klammern geändert werden.

Klammern verändern die Reihenfolge

Arithmetische Operatoren gleicher Priorität werden von links nach rechts zusammengefaßt (Zusammenfassung = Assoziativität). Als Beispiel die Auswertung von 10*10/5:

```
10*10/5 = 100/5 = 50
```

Damit Sie die Werte Ihrer Variablen nach einer arithmetischen Operation untereinander vergleichen können, stellt C sechs Vergleichsoperatoren zur Verfügung. Vergleichsoperatoren sind immer binäre Operatoren, da ja zwei Werte verglichen werden:

»<« — kleiner als. Beispiel: x<y ergibt 1, wenn x kleiner ist als y, ansonsten ist der Wert 0.

»>« — größer als. Beispiel: x>y ergibt 1, wenn x größer als y ist, oder 0, wenn diese Aussage nicht zutrifft.

»<=« — kleiner gleich. x<=y ergibt 1, wenn x höchstens so groß wie y ist, ansonsten 0.

»>=« — größer gleich. x>=y liefert den Wert 1, wenn x mindestens den Wert von y hat, sonst 0.

»==« — gleich. x==y ergibt 1, wenn x genau y ist und 0, wenn die Werte verschieden sind.

»!=« — ungleich. x!=y ergibt 1, wenn x einen von y verschiedenen Wert hat. Sind x und y gleich, liefert dieser Ausdruck 0.

```
110 /*
120       zahlen - Operationen mit
          Zahlen
130 */
140
150 char *text_1[]={
160      "Die Zahl ist kleiner als
         10 ",
170      "Ihre Zahl ist zu groß "
180 };
190
200 char *text_2[]={
210      "und sie ist größer als
         0.",
220      "aber sie ist zu klein!"
230 };
240
250 void main()
260 {
270
280      int zahl, bool;
290
300      printf("\nBitte geben Sie
         eine Zahl zwischen 0 und
         10 ein! > ");
310      scanf("%d",&zahl);
320
330      bool = zahl>=10;
340      printf("%s",text_1[bool]);
350
360      bool = zahl<=0;
370      printf("%s\n",text_2[bool
         ]);
380
390      printf("%d * %d = %d\n",z
         ahl,zahl,zahl*zahl);
400      printf("%d %% 3 = %d\n\n"
         ,zahl,zahl%3);
410 }
```

In diesem Programm entscheidet der Wert der Variablen »bool«, welcher Text auf dem Bildschirm ausgegeben wird. Die Variable erhält ihren Wert aus dem Vergleich zweier Zahlen. Ist die Aussage hinter dem Zuweisungsopator wahr, so wird »bool« auf 1 gesetzt.

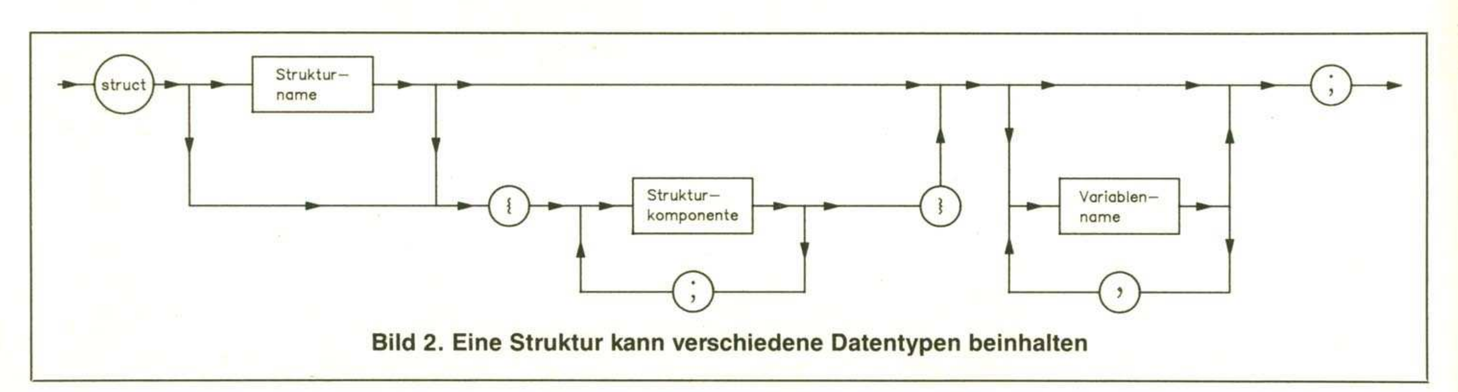


Bild 2. Eine Struktur kann verschiedene Datentypen beinhalten

Wenn die Aussage nicht zutrifft, erhält »bool« den Wert 0.

Man kann diese booleschen Werte (wahr/falsch oder TRUE/FALSE) auch miteinander verknüpfen. Operatoren für logische Verknüpfungen sind:

»!« — logische Negation (unitär). Die Negation erzeugt entweder den Wert 0, wenn der Operand einen Wert ungleich 0 liefern würde oder 1, wenn er den Wert 0 liefern würde.

```
110 int bool, zahl=10;
120
130 bool = !(zahl>100);
```

Die Variable »bool« erhält den Wert 1, denn NICHT 0 ergibt 1!

»&&« — logische UND-Verknüpfung (binär). Merksatz: Die UND-Verknüpfung liefert den Wert 1, wenn der eine UND der andere Operand einen von 0 verschiedenen Wert hat.

```
0 && 0 = 0
0 &&11 = 0
1 && 0 = 0
1 && 1 = 1
```

»||« — die logische ODER-Verknüpfung (binär). Auch hier ein Merksatz: Die ODER-Verknüpfung liefert den Wert 1, wenn einer ODER beide Operanden einen Wert ungleich 0 haben.

```
0 || 0 = 0
0 || 1 = 1
1 || 0 = 1
1 || 1 = 1
```

Bit für Bit

Aber auch einzelne Bits können in C logisch verknüpft werden. Hierfür stehen uns wiederum sechs Operatoren zur Verfügung:

»&« — UND-Verknüpfung von Einzelbits (binär). Erinnern Sie sich an den Merksatz:

```
10101010 & 10100101 =
10100000
```

Im Ergebnis wird nur an der Stelle ein Bit gesetzt, wo die Bits beider Operanden ebenfalls auf 1 gesetzt sind.

»|« — ODER-Verknüpfung von Einzelbits (binär).

```
10101010 | 10100101 =
10101111
```

Wenn eines oder gar beide Bits der Operanden gesetzt sind, wird das jeweilige Bit auch im Ergebnis gesetzt.

»^« Exclusiv-ODER-Verknüpfung (binär). Im Ergebnis wird ein Bit gesetzt, wenn genau eines der jeweiligen Bits der Operanden gesetzt ist:

```
10101010 ^ 10100101 =
00001111
```

»~« Bit-Komplement (unitär). Wenn ein Bit des Operanden gesetzt ist, so wird es gelöscht und umgekehrt:

```
~ 10101010 = 01010101
```

»>>« und »<<« — Rechts- beziehungsweise Linksverschieben (shiften) von Bitfolgen (binär). Der Operand links vom Operator enthält das zu verschiebende Bitmuster. Der rechte Operand gibt an, um wie viele Positionen das Muster verschoben werden soll. Angenommen der Wert einer Variablen hätte das Bitmuster 00111100:

```
00111100>>2 =
00001111
oder
00111100<<2 =
11110000
```

Sie sehen bei einer Verschiebung nach links oder rechts werden immer Nullen nachgezogen.

Um den Wert einer Variablen zu ändern, haben Sie bisher nur den Zuweisungsoperator »=« kennengelernt:

```
variable = neuer_Wert;
```

Dank dem kreativen Schaffen der Sprachdesigner von C, Kernighan und Ritchie, warten aber noch zehn weitere Zuweisungsoperatoren auf ihren Einsatz!

Alle Zuweisungsoperatoren haben die gleiche Priorität. Eine Auflistung finden Sie in Tabelle 3, zweite Zeile von unten.

Wie diese Operatoren gehandhabt werden, wird am Beispiel »+=« erläutert. Alle anderen, abgesehen von dem Gleichheitszeichen, dessen Funktion Sie ja bereits kennen, funktionieren analog hierzu.

Angenommen, Sie möchten im Programm den Wert der Variablen x um 5 erhöhen. Eine Möglichkeit, die zum Beispiel Basic bietet ist:

```
x = x + 5
```

Solche Ausdrücke dürfen in C wie folgt abgekürzt werden:

```
110 int x=10;
120
130 x+=5;
    /* entspricht
    x=x+5 */
```

Nach der Zuweisung in Zeile 130 hat x den Wert 15. Der Wert der Variablen x wurde mit 5 inkrementiert (erhöht). Inkrementierungen und Dekrementierungen (Verminderungen) um 1 können sogar noch kürzer geschrieben werden:

```
150 x++;
160 x--;
```

In Zeile 150 wird zum Wert der Variablen »x« 1 addiert und in Zeile 160 gleich wieder 1 subtrahiert. Auch folgende Zeilen sind erlaubt:

```
170 ++x;
180 --x;
```

Im ersten Beispiel (Zeilen 150 und 160) wird zuerst mit der Variablen gearbeitet. Erst dann wird deren Wert verändert. Diese Art der Verwendung der Inkrement- und Dekrementoperatoren (unitär) wird Postfix-Notation genannt. Die Zeilen 170 und 180 zeigen eine Präfix-Notation. Zuerst wird der Wert der Variablen geändert, dann wird sie benutzt. Der Unterschied wird im folgenden Beispiel deutlich:

```
100
110 int a, b=10;
120
130 a=b++;        /* a erhält den
    Wert 10 und anschließend wird
    b erhöht */
140 a=b;          /* a wird der Wert
    von b (11) zugewiesen */
150 a=++b;        /* der Wert von a
    ist nun 12, wie der von b */
160 --b;          /* b = 11 */
170 b--;          /* b = 10;  hier
    ist kein Unterschied zu Zeile
    160 */
180
190 printf("%d",b++); /* der Wert
    10 wird ausgegeben, b wird er
    höht */
200 printf("%d",b);   /* b hat den
    Wert 11 */
210 printf("%d",++b); /* der Wert
    von b (12) wird ausgegeben */
220
```

SCHALTER	BIT 1	Bit 2
0	0	0
1	0	1
2	1	0
3	1	1

Tabelle 1. Ein Bitfeld als Schalter mit vier Stellungen

Amigadatentypen	Standard C-Typen
LONG	long
ULONG	unsigned long
LONGBITS	unsigned long
WORD	short
UWORD	unsigned short
WORDBITS	unsigned short
BYTE	char
UBYTE	unsigned char
BYTEBITS	unsigned char
STRPTR	*unsigned char
APTR	*unsigned char
SHORT	short
USHORT	unsigned short
FLOAT	float
DOUBLE	double
COUNT	short
UCOUNT	unsigned short
BOOL	short
TEXT	unsigned char

Tabelle 2. Die in der Headerdatei »exec/types.h« definierten Amiga-Datentypen, die bevorzugt werden sollten

Operator	Assoziativität
() [] -> .	von links nach rechts
! ~++ — * & sizeof (TYP)	rechts nach links
* / %	links nach rechts
+ -	links nach rechts
« »	links nach rechts
< <= > >=	links nach rechts
== !=	links nach rechts
&	links nach rechts
^	links nach rechts
|	links nach rechts
&&	links nach rechts
||	links nach rechts
?:	rechts nach links
= += -= /= *= %= »= «=	
&= |= ^=	rechts nach links
,	links nach rechts

Tabelle 3. Die Operatoren der Programmiersprache C, nach Prioritäten geordnet, und ihre Assoziativität

Die letzten fünf Operatoren lassen sich nicht oder nur schlecht einordnen. Die meisten davon kennen Sie bereits:

»[]« — die eckigen Klammern nennt man Indexoperator oder auch Arrayselektor.

»(Datentyp)« — mit Hilfe des Cast-Operator (unitär) können Datentypen konvertiert werden.

»sizeof()« — der Sizeof-Operator liefert die Größe des in Klammern folgenden Datentyps oder Ausdrucks, in Bytes.

»*« — über den Inhaltsoperator (unitär) gelangt man zu dem Wert einer Zeigervariablen.

»&« — der Adreßoperator (unitär) ermittelt die Adresse einer Variablen.

»,« — Der Kommaoperator (binär) trennt Ausdrücke, welche dann von links nach rechts abgearbeitet werden. Der Kommaoperator, zusammen mit den Operanden, muß in Klammern geschrieben werden, wenn er in einer Argumentenliste oder in einer Liste von Variablennamen eingesetzt werden soll:

```
100 int z;
110 ...
120 printf("Die Variable
    z hat den Wert %d",
    (z=10,z+10));
```

Es wird zuerst der Ausdruck »z=10« und danach »z+10« abgearbeitet.

Folglich wird als Wert der Variablen 20 ausgegeben.

»?:« — die bedingte Bewertung, der letzte Operator der Programmiersprache C benötigt als einziger drei Operanden und ist somit ein ternärer Operator.

```
Operand1 ? Operand2 :
Operand3
```

Operand1 wird zuerst ausgewertet. Ist das Ergebnis von 0 verschieden (wahr), so wird der Wert von Operand2 zurückgegeben, ansonsten der Wert von Operand3:

```
ergebnis = flag==0 ? 10 :
 -10;
```

Wenn der Wert der Variablen »flag« gleich 0 ist, wird der Variablen »ergebnis« der Wert 10, im anderen Fall -10 zugeteilt.

Somit sind wir am Ende des dritten Teils unseres C-Kurses angelangt und wie immer an dieser Stelle gibt es noch einige Übungen, damit Ihnen die Zeit bis zur nächsten Ausgabe nicht zu lange wird:

1) Initialisieren Sie eine Matrix mit verschiedenen Werten. Multiplizieren Sie diese Matrix mit einem int-Wert, den Sie von der Tastatur lesen. Geben Sie die Werte auf dem Bildschirm aus.

2) Definieren Sie eine Strukur, die Ihre persönlichen Daten und das aktuelle Datum aufnehmen kann. Definieren Sie Variablennamen und -zeiger dieses Typs. Belegen Sie die Strukturkomponenten mit Werten, verändern Sie sie, geben Sie sie auf dem Bildschirm aus.

3) Setzen und löschen Sie einzelne Bits eines int-Wertes. Geben Sie ihn, zur Kontrolle, nach jeder Operation auf dem Bildschirm aus.

4) Lesen Sie einen int-Wert über die Tastatur ein. Shiften Sie ihn nach links und rechts. Geben Sie jeweils das Ergebnis auf dem Bildschirm aus. Wie verhält es sich zu der eingegebenen Zahl?

5) Schreiben Sie ein Programm, das mit Hilfe der bedingten Bewertung den Absolutwert einer über die Tastatur eingegebene Zahl ausgibt.

(Arno Gölzer/rb)

```
110 /*
120      Pref - einige Werte der Preferences-Struktur ausgeben
130 */
140
150 #include <intuition/intuition.h>
160
170 struct IntuitionBase *IntuitionBase; /* Zeiger IntuitionBase de-
finieren */
180 struct Preferences Preferences; /* Strukturname Preferences defi-
nieren */
190
200 char *cli[]={ /* Zeiger-Arrays beinhalten die Zustände des CLI-
Icon */
210      "unsichtbar",
220      "sichtbar"
230 };
240 char *papier[]={ /* Zeiger-Arrays beinhalten die Papiertypen */
250      "Endlosblatt",
260      "Einzelblatt"
270};
280 char *schrift[]={ /* Zeiger-Arrays beinhalten die Schriftarten */
290      "Draft",
300      "Letter"
310 };
320
330 /*
340       Funktionen
350 */
360
370 void start()
380 /*
390     Ein Library wird geöffnet;
400     Die Programmüberschrift wird ausgegeben;
410 */
420 {
430      /* intuition.library öffnen */
440      if(!(IntuitionBase = OpenLibrary("intuition.libra
     ry",0L))) exit();
450      printf("\f\n\n\033[3m\033[4mEinige Wer
     te aus PREFERENCES\033[0m\n\n\n");
460 }
470
480 void ende()
490 /*
500    Das Library wird geschlossen;
510    Programmende;
520 */
530 {
540      CloseLibrary(IntuitionBase); /* intuition.library schlie
     ßen */
550      exit(); /* Programmende */
560 }
570
580 void main()
590 {
600    int element; /* Variable die die Nummer eines Arrayelemen
   tes aufnimmt */
610
620  start();     /* Funktionsaufruf */
630  /* Intuitionfunktion GetPrefs(): Preferences-Struktur lesen.
640     Parameter sind: -Zeiger auf Speicherbereich, der durch die
     Struktur
650                  belegt werden soll (&Parameter)
660                 -Anzahl der Bytes, die gelesen werden sollen
670                  (sizeof(structPreferences) = die gan
                 ze Struktur);
680  */
690  GetPrefs(&Preferences,sizeof(struct Preferences));
700
710  /* einige Preferences-Werte ausgeben: */
720  printf("Allgemeines:\n\n");
730  printf("Zeichenhöhe: %d\n",Preferences.FontHeight);
740  /* Zugriff auf die DoubleClick-Struktur in der Preferences
  Struktur */
750  printf("Zeit für Doppelklick: ca. %ld Sek.\n",Preferences.Dou
  bleClick.tv_secs);
760  /* EnableCLI kann den Wert 0 oder 1 annehmen und gibt hier an,
  welches
770     Zeiger-Array (cli[0] oder cli[1]) ausgegeben werden soll */
780  printf("CLI-Icon: %s\n\n\n",cli[Preferences.EnableCLI]);
790
800  printf("Druckereinstellungen:\n\n");
810  printf("Abstand vom linken Rand: %d Zeichen\n",Preferences.
  PrintLeftMargin);
820  printf("Abstand vom rechten Rand: %d Zeichen\n",Preferences.
  PrintRightMargin);
830  printf("Zeilen pro Seite: %d\n",Preferences.PaperLength);
840  /* Wenn PaperType den Wert 0 hat, erhält element ebenfalls 0.
850     Ansonsten wird element der Wert 1 zugewiesen. */
860  element = (Preferences.PaperType==0) ? 0 : 1;
870  printf("Papierart: %s\n",papier[element]);
880  /* Wenn PrintQuality den Wert 0 hat, erhält element eben
  falls 0.
890     Ansonsten wird element der Wert 1 zugewiesen. */
900  element = (Preferences.PrintQuality==0) ? 0 : 1;
910  printf("Schriftqualität: %s\n\n\n",schrift[element]);
920
930  ende();
940 } /* main */
```

Listing 1. Mit »Pref« wird die aktuelle Preferences-Struktur der Systemdatei ausgelesen und einige Werte davon auf dem Bildschirm ausgegeben

Basic für Aufsteiger

Möchten Sie Zeit beim Schreiben und Austesten Ihrer Programme sparen? Dann nutzen Sie Unterprogramme. Amiga-Basic bietet neben den bekannten Techniken echte Unterprogramme und eine komfortable Einbindung von Programmroutinen in Maschinensprache.

Im letzten Teil unseres Basic-Kurses beschäftigen wir uns mit anspruchsvollen Themen der Basic-Programmierung. Dazu gehören sowohl die Verwendung von Unterprogrammen als auch der Aufruf eines Maschinenprogramms von Basic aus. Außerdem erfahren Sie, wie sich die im Betriebssystem des Amiga vorhandenen Libraries von einem Basic-Programm aus nutzen lassen.

Vielleicht ist Ihnen noch nicht aufgefallen, daß wir bereits im vierten Teil unseres Kurses mit Unterprogrammen gearbeitet haben. Die Arbeitsweise der Unterbrechungsverarbeitung beruht im wesentlichen auf der Unterprogrammtechnik. Wenn Sie den letzten Teil des Kurses aufmerksam verfolgt haben, können Sie die Vorteile und Einsatzmöglichkeiten der Unterprogrammtechnik bereits abschätzen: Programme lassen sich durch Auslagerung logisch zusammengehöriger Teile übersichtlicher gestalten. Eine solche Methodik unterstützt die im Kapitel über die strukturierte Programmierung empfohlene Technik des Top-Down-Design (siehe Teil 3 dieses Kurses).

Einen gewichtigen Nachteil der bisher dargestellten Unterprogrammtechnik dürfen wir natürlich nicht verschweigen. Unterprogramme, die mit Hilfe des GOSUB-Befehles aufgerufen und mit dem RETURN-Befehl beendet werden, sind eigentlich keine echten Unterprogramme. Eine derart schwere Anschuldigung bedarf selbstverständlich einer Rechtfertigung: Ein Unterprogramm sollte von dem Programm, das es aufruft, so unabhängig wie nur irgend möglich sein. Dies ist notwendig, damit es einerseits auch in anderen Programmen sinnvoll eingesetzt und andererseits ohne Anpassungsschwierigkeiten durch ein neues, besseres Unterprogramm ersetzt werden kann.

Die Unabhängigkeit vom »Hauptprogramm« ist allerdings bei diesen mit GOSUB aufgerufenen Routinen nicht erfüllt, denn alle Variablen des Unterprogramms sind auch im übergeordneten Programm bekannt und umgekehrt. Das hat folgende Konsequenzen:

Erstens: Bei der Entwicklung des Unterprogramms muß der Programmierer darauf achten, daß er nicht unbeabsichtigt Variablennamen verwendet, die auch schon im Hauptprogramm benutzt werden. Das kann fatale Folgen haben: ein Unterprogramm ändert Variablenwerte, die im Hauptprogramm für eine andere Aufgabe vorgesehen sind. Die Fehlersuche in solchen Fällen ist in der Regel recht langwierig.

Zweitens: Die Variablenübergabe an solche Unterprogramme ist umständlich. Ein einfaches Beispiel: Mit dem Unterprogramm

```
LOCATE Zeile%,Spalte%
PRINT Text$
RETURN
```

lassen sich beliebige Texte an eine beliebige Stelle des Bildschirms ausgeben. Vor dem Aufruf der Routine müssen den Variablen Zeile%, Spalte% und Text$ natürlich noch die entsprechenden Werte zugewiesen werden.

Original und Fälschung

Um diesen Umstand zu beseitigen, haben die Entwickler des Amiga-Basic eine der wenigen Basic-Dialekte geschaffen, der auch echte Unterprogramme mit »lokalen« Variablen kennt. Die Bezeichnung »lokal« bedeutet in diesem Zusammenhang, daß diese Variablen nur dem Unterprogramm, nicht jedoch dem Hauptprogramm bekannt sind. So werden Namenskonflikte vermieden. Sie können sowohl in Ihrem Hauptprogramm, als auch im Unterprogramm eine Variable mit dem Namen »x« definieren. Beides sind zwei verschiedene Variablen. Während »x« im Hauptprogramm beispielsweise den Wert Null hat, kann »x« im Unterprogramm Eins enthalten.

Mit der Verwendung lokaler Variablen können Sie sich eine Sammlung oft benötigter Unterprogramme aufbauen und einzelne Routinen später je nach Bedarf in Ihre Programme einfügen, ohne darauf achten zu müssen, ob diese Variablen mit gleichnamigen im Hauptprogramm kollidieren. Ohne eine solche Arbeitsweise wäre den Profis unter den Programmierern ein ökonomisches Arbeiten nicht mehr möglich.

Wie bringen Sie nun Ihrem Amiga bei, daß Sie fortan nur noch »echte« Unterprogramme schreiben möchten? Sicherlich genügt nicht mehr die Angabe einer Sprungmarke wie bei den mit GOSUB aufgerufenen Unterprogrammen. Ein echtes Unterprogramm wird mit der Anweisung

```
SUB Unterprogrammname
(Parameterliste) STATIC
```

begonnen. Obwohl die Variablen echter Unterprogramme in der Regel nichts mit denen des Hauptprogrammes zu tun haben, müssen auf irgend eine Weise die zu verarbeitenden Werte übergeben werden. Dazu dient die Parameterliste. Sie enthält eine Aufzählung formaler Variablen, an die die von Ihnen gewünschten Werte übergeben werden können. Der Name »formale Variablen« rührt von der Tatsache her, daß diesen Variablen im Unterprogramm nur insofern Bedeutung zukommt, als sie als Platzhalter für die dann im Unterprogrammaufruf tatsächlich übergebenen aktuellen Variablen dienen, mit denen die Berechnungen dann durchgeführt werden.

Entsprechend der RETURN-Anweisung bei den unechten Unterprogrammen bewirkt bei den echten Unterprogrammen die Anweisung

```
END SUB
```

das Beenden des Unterprogramms. Genau wie bei ihren Gegenstücken können echte Unterprogramme mit

```
EXIT SUB
```

von beliebiger Stelle innerhalb der Routine verlassen werden. Wenn wir schon dauernd davon sprechen, ein Unterprogramm »aufzurufen« — was liegt also näher als den zugehörigen Befehl CALL zu nennen. Mit

```
CALL Unterprogrammname
(Parameterliste)
```

veranlassen Sie Ihren Computer, die Programmausführung mit dem genannten Unterprogramm fortzuführen. Die formalen Parameter des Unterprogramms erhalten die Werte der in dieser Parameterliste aufgeführten aktuellen Parameter. Wenn keine Verwechslungen (etwa mit Sprungmarken) zu befürchten sind, können Sie den CALL-Befehl auch weglassen. Ebenso können Sie darauf verzichten, die Parameterliste in Klammern einzuschließen.

Lokale Variablen sind nicht in jedem Fall erwünscht. Um dem Programmierer eine Möglichkeit zu geben, Variablen des Hauptprogramms vom Unterprogramm bewußt zu verändern, besitzt Amiga-Basic die

Teil 5

KURSÜBERSICHT

Dieser Kurs hat das Ziel, Ihnen alles über die Programmierung des Amiga-Basic beizubringen. In den einzelnen Teilen des Kurses finden Sie folgende Themen behandelt:

TEIL 1 (Grafik): Zeichnen von Punkten, Linien und Mustern; Verschieben von Bildschirmausschnitten

TEIL 2 (Animation/Sound): Bewegung von Objekten auf dem Bildschirm; Soundunterstützung

TEIL 3 (Strukturierte Programmierung): Umgang mit IF.. THEN..ELSE und WHILE.. WEND; optische Gestaltung von Quelltexten

TEIL 4 (Unterbrechungsverarbeitung): Abfangen von Programmfehlern; Ablaufunterbrechungen durch Menü-, Timer- oder Mausereignisse.

TEIL 5 (Unterprogramme/ Libraries): Unterprogrammtechnik; Einbinden von Maschinenprogrammen und Libraries

SHARED-Anweisung. Mit dem Attribut »shared« werden Variablen zu globalen, also vom ganzen Programm verfügbare, Variablen. Wenn Sie in Ihrem Unterprogramm (am besten gleich nach der SUB-Anweisung) den Befehl

```
SHARED Variablenliste
```

angeben, so besitzen die genannten Variablen im Unterprogramm denselben Wert wie im Hauptprogramm.

Listing 1 soll Ihnen dazu dienen, die bisher gewonnenen Kenntnisse an einem praktischen Beispiel zu erproben. Vor dem Lauf sollten Sie in einem »Schreibtischtest« ermitteln, welche Werte auf dem Bildschirm ausgegeben werden. Gelingt Ihnen dies, haben Sie das Prinzip der lokalen und globalen Variablen bereits verstanden.

Assembler in Basic

Nun wird es aber Zeit, daß wir uns einem komplizierten Thema zuwenden: dem Einbinden von Maschinenprogrammen in Basic. Wir halten uns im wesentlichen an das in Ihrem Basic-Handbuch in Kapitel 6.2 angegebene Beispiel. Das dort abgedruckte Programm enthält allerdings einen Druckfehler und zwei Ungenauigkeiten, die in der hier in Listing 2 abgedruckten Version behoben sind. Aufgabe dieses Programms ist es, in einer beliebigen Zeichenkette alle Kleinbuchstaben in Großbuchstaben umzuwandeln. Da das im Basic-Handbuch angegebene Assemblerprogramm ausreichend kommentiert ist, können wir uns hier eine weitere Dokumentation sparen. Wir gehen davon aus, daß Ihnen ein Assembler zur Verfügung steht, um das Programm in ausführbaren Maschinencode zu übersetzen.

Das in Listing 2 abgedruckte Programm wurde mit dem zum Lattice-C-Compiler 3.10 mitgelieferten Makro-Assembler »asm« übersetzt und danach mit dem Programm »blink« gelinkt. Einen Hexdump des erzeugten, ausführbaren Files sehen Sie in Bild 1. Was sofort auffällt, ist eine gewisse Diskrepanz in der Dateilänge zur Länge des eigentlichen Maschinenprogramms. Während des Assemblierens wurde die Länge des Maschinenprogramms mit 56 Byte angegeben. Das vom Linker erzeugte File ist aber 92 Byte lang. Mit Hilfe dieser Zusatzinformationen kann der Lader des Amiga Adreßumrechnungen vornehmen. Da dies aber nur notwendig ist, wenn im Assemblerprogramm absolute Adressen verwendet werden — und dies ist in unserem Beispiel nicht der Fall — können wir diese Informationen überlesen.

Die für uns interessante Stelle wird durch die Hex-Zahl $03E9 gekennzeichnet. Diese Zahl teilt Ihrem Amiga mit, daß das nächste Langwort (4 Byte) die Länge des Maschinencodes in Langworten enthält und der Maschinencode selbst im übernächsten Langwort beginnt. Damit ist unsere Strategie zum Einlesen des Maschinencodes klar vorgezeichnet: Das gesamte vom Linker erzeugte File wird in eine Stringvariable gelesen und die Hex-Zahl $000003E9 gesucht. Dann wird die Länge des Maschinencodes bestimmt und dieser in eine neue Stringvariable eingelesen. Hier muß allerdings ein kleiner Trick angewendet werden: Es scheint, als ob Strings, die eine gerade Anzahl von Bytes enthalten, immer bei einer ungeraden Adresse beginnen. Maschinenprogramme müssen dagegen immer bei einer geraden Adresse starten. Daher ist es unbedingt notwendig, daß der String, der das Maschinenprogramm enthält, eine ungerade Länge erhält — wir addieren zur errechneten Stringlänge eine 1. Das in Listing 3 angegebene Unterprogramm »ReadCode« führt genau diese Aktionen aus. Eine Einschränkung besteht noch für die hier angegebene Methode, Maschinenprogramme einzulesen: Da Strings nicht mehr als 32 KByte Zeichen enthalten können, darf auch das Maschinenprogramm nicht länger als 32 KByte sein.

```
0000: 000003F3 00000000 00000001 00000000   ................
0010: 00000000 0000000E 000003E9 0000000E   ................
0020: 48E7C080 202F0010 206F0014 42816000   H... /.. o..B.`.
0030: 001C1230 00000C01 00616D00 00100C01   ...0.....am.....
0040: 007A6E00 00080230 00DF0000 51C8FFE4   .zn....0....Q...
0050: 4CDF0103 4E750000 000003F2            L...Nu......
```

Bild 1. Hexdump des fertigen Programms

Wir haben das Maschinenprogramm nun gelesen. Wie wird es aber nun zur Ausführung gebracht? Hier hilft uns wieder der CALL-Befehl. Nach einer CALL-Anweisung muß nicht unbedingt ein Unterprogrammname stehen, sondern es kann ebenso eine Variable angegeben werden, die die Anfangsadresse eines Maschinenprogramms enthält.

Parameter auf den Stapel

Diese Adresse erhalten Sie mit der Funktion SADD. Die von dem Unterprogramm benötigten Parameter sind wie gewohnt innerhalb der Klammern anzuführen. Sie werden auf den Stack abgelegt, von wo sie sich das Maschinenprogramm abholen kann. Auch hier gibt es eine Einschränkung zu beachten: Es können nur kurze und lange Ganzzahlen übergeben werden. Von den anderen Variablentypen (Arrays, Strings...) müssen Sie die Adresse übergeben. Diese läßt sich einfach mit VARPTR oder SADD ermitteln. Wenn Sie beim Aufruf von Maschinenprogrammen von Basic aus Systemabstürze vermeiden wollen, müssen Sie unbedingt eine Eigenart von Basic berücksichtigen: Sobald Sie im laufenden Programm eine neue Variable benutzen, werden sich die Anfangsadressen aller Strings ändern. Sie sollten daher die Startadresse Ihres Maschinenprogramms erst unmittelbar vor dem Unterprogrammaufruf ermitteln. Listing 3 gibt Ihnen auch hierzu ein Beispiel.

Vielleicht haben Sie keinen Assembler und ärgern sich jetzt, weil wir Ihnen etwas schmackhaft gemacht haben,

```
a = 1 : b = 2 : c = 3
x = 4 : y = 5 : z = 6
PRINT a,b,c,x,y,z : PRINT
GOSUB pseudo      : PRINT
CALL  echt(a,b,z) : PRINT
GOSUB pseudo      : PRINT
x = 400 : y = 500 : c = 300
GOSUB pseudo      : PRINT
END

pseudo:
  PRINT a,b,c,x,y,z
RETURN

SUB echt(x,y,z) STATIC
   SHARED a,c
   b = 20
   z = 30
   PRINT a,b,c,x,y,z
   a = 10
END SUB
```

Listing 1. Lokale und globale Variable

```
        CSECT CODE
        movem.l a0/d0-d1,-(sp)
        move.l  16(sp),d0
        move.l  20(sp),a0
        clr.l   d1
        bra     test
start:
        move.b  0(a0,d0.w),d1
        cmp.b   #$61,d1
        blt     test
        cmp.b   #$7a,d1
        bgt     test
        and.b   #($ff-$20),0(a0,d0.w)
test:
        dbf     d0,start
        movem.l (sp)+,a0/d0-d1
        rts
        END
```

Listing 2. Assemblercode zur Stringumwandlung

```
##base _MathTransBase
##bias 30
##public
SPAtan(float)(D0)
SPSin(float)(D0)
SPCos(float)(D0)
SPTan(float)(D0)
SPSincos(leftFloat,rightFloat)(D1,D0)
SPSinh(float)(D0)
SPCosh(float)(D0)
SPTanh(float)(D0)
SPExp(float)(D0)
SPLog(float)(D0)
SPPow(leftFloat,rightFloat)(D1,D0)
SPSqrt(float)(D0)
SPTieee(float)(D0)
SPFieee(integer)(D0)
*-----------------------------------------
*
*  New functions added for Release 1.1
*
*-----------------------------------------
SPAsin(float)(D0)
SPAcos(float)(D0)
SPLog10(float)(D0)
##end
```

Bild 2. Die FD-Dateien enthalten die Funktionsnamen und Parameterinformationen einer Bibliothek

```
0000: 53504174 616E00FF E2010053 5053696E    SPAtan.....SPSin
0010: 00FFDC01 00535043 6F7300FF D6010053    .....SPCos.....S
0020: 5054616E 00FFD001 00535053 696E636F    PTan.....SPSinco
0030: 7300FFCA 02010053 5053696E 6800FFC4    s......SPSinh...
0040: 01005350 436F7368 00FFBE01 00535054    ..SPCosh.....SPT
0050: 616E6800 FFB80100 53504578 7000FFB2    anh.....SPExp...
0060: 01005350 4C6F6700 FFAC0100 5350506F    ..SPLog.....SPPo
0070: 7700FFA6 02010053 50537172 7400FFA0    w......SPSqrt...
0080: 01005350 54696565 6500FF9A 01005350    ..SPTieee.....SP
0090: 46696565 6500FF94 01005350 4173696E    Fieee.....SPAsin
00A0: 00FF8E01 00535041 636F7300 FF880100    .....SPAcos.....
00B0: 53504C6F 67313000 FF820100             SPLog10.....    .
```

Bild 3. Ein Hexdump der von ConvertFD in das ».bmap«-Format umgewandelten FD-Datei

was Sie nicht realisieren können. Keine Angst — zur Not und mit etwas Handarbeit geht es auch ohne Assembler.

Jedem in Listing 2 angegebenen Assemblerbefehl entspricht (mindestens) eine kurze Ganzzahl. Die unserem Maschinenprogramm entsprechenden Zahlen sehen Sie im Assemblerlisting in Ihrem Basic-Handbuch am linken Rand. Diese Zahlen können Sie in Ihrem Basic-Programm genausogut in DATA-Zeilen schreiben und diese dann mit Hilfe des READ-Befehles lesen. In diesem Fall müssen Sie allerdings für die Dimensionierung im voraus wissen, wie lang das Maschinenprogramm ist.

Bibliotheken

Kennt Ihr Basic-Programm erst einmal die Anzahl der Maschinenbefehle, so kann ein Ganzzahlfeld mit derselben Anzahl dimensioniert werden. In dieses Feld muß nun das Maschinenprogramm eingelesen werden. Die Anfangsadresse erhalten Sie durch die Anwendung der VARPTR-Funktion auf das erste Feldelement (Listing 4).

Die Organisation des Betriebssystems in Libraries (Bibliotheken) sichert einen einfachen Zugriff auf die Routinen der Systemsoftware. Die Softwareentwickler des Amiga haben aber nicht nur für das Betriebssystem, sondern auch für Programmsammlungen, die man nicht direkt dem Betriebssystem zuordnen kann, diese Organisationsform gewählt. Ein Beispiel dafür ist eine Library zur Berechnung der auch als transzendente Funktionen bezeichneten Winkelfunktionen (Sinus, Cosinus, Tangens...). Wir zeigen Ihnen, wie Sie diese Bibliothek auch von Basic aus nutzen können.

Leider können Sie die Library-Funktionen nicht direkt ansprechen, da sich die Parameterübergabe von Basic von der der Libraries unterscheidet. Deshalb müssen vorher noch einige Vorbereitungen getroffen werden. Um die Library »mathtrans.library« benutzen zu können, müssen Sie dem Interpreter erst mitteilen, welche Funktionen in dieser Library vorhanden sind und in welchem Prozessorregister die Parameter übergeben werden sollen. Auf Ihrer Extras-Diskette finden Sie eine Schublade mit dem Namen FD1.2 (falls Sie Kickstart 1.2 besitzen). Um sich die in dieser Schublade enthaltenen Files betrachten zu können, müssen Sie allerdings den CLI-Editor »ed« verwenden. Das für uns interessante File heißt »mathtrans_lib.fd« (siehe Bild 2). In diesem File wird genau beschrieben, welche Funktionen in der Bibliothek vorhanden sind und in welchen Registern die Parameter übergeben werden müssen. Amiga-Basic kann mit der Form der Daten in dieser Datei allerdings nichts anfangen. Deshalb befindet sich in der Schublade »BasicDemos« das Programm ConvertFD, welches die Daten der FD-Datei in ein für Basic verständliches Format umwandelt. Die vom Programm generierte Datei heißt »mathtrans.bmap«.

Die Werte der ».bmap«-Datei können von Amiga-Basic einfacher verarbeitet werden. Eine vorherige Umwandlung erspart dem Interpreter die zeitaufwendige Interpretation der Daten aus der FD-Datei beim Programmablauf.

Wie sieht nun das Format eines solchen .bmap-Files aus? Dazu sollten wir uns erst einmal Überlegen, welche Informationen der Basic-Interpreter eigentlich benötigt, um die jeweilige Funktion aufrufen zu können. Dies wäre zunächst einmal der Name der Funktion selbst. Dann muß die Sprungadresse, also die Stelle, an der sich die Routine innerhalb der Library befindet, bekannt sein. Schließlich muß der Interpreter noch wissen, welche Parameter die Funktion benötigt und in welchen der Prozessorregister diese übergeben werden.

Genau diese Informationen stehen in der .bmap-Datei. Für jede Funktion — und eine Funktion ist ja nichts anderes als ein Unterprogramm — steht dort der von einem Nullbyte abgeschlossene Funktionsname als ASCII-Code, gefolgt von der Position dieser Funktion in der Library. Dabei wird die Position relativ zum Anfang der Library als 16-Bit-Offset ausgedrückt. (Ein Offset ist ein Wert, der auf eine Basisadresse — in diesem Fall derjenigen der Library — aufaddiert werden muß.) Danach folgt für jedes Register, in dem ein Wert übergeben wird, 1 Byte. Dieses Byte enthält eine Codezahl, die

```
ReadCode "Code",McProg$
PRINT "Geben Sie einen String ein: "
INPUT st$
UPPER st$
PRINT st$
END

SUB UPPER(argument$) STATIC
  SHARED McProg$
  adr&   = SADD(argument$)
  laeng& = LEN(argument$)
  entry& = SADD(McProg$)
  CALL entry&(laeng&,adr&)
END SUB

SUB ReadCode(filename$,AsmCode$) STATIC
  OPEN filename$ FOR INPUT AS #1
  file$ = INPUT$(LOF(1),#1)
  CLOSE #1
  HunkCode&  = &H3E9
  i          = 1
  code&      = CVL(MID$(file$,i,4))
  WHILE (code& <> HunkCode&)
    i     = i + 4 : REM Nächstes Langwort
    code& = CVL(MID$(file$,i,4))
  WEND
  CodeLength& = 4*CVL(MID$(file$,i+4,4))+1
  AsmCode$ = MID$(file$,i+8,CodeLength&)
END SUB
```

Listing 3. Maschinenprogrammaufruf in Basic

angibt, welches Register betroffen ist. Dabei stehen die Zahlen 1 bis 8 für die Register D0 bis D7, die Zahlen 9 bis 13 stehen für die Register A0 bis A4. Nach der letzten Registerangabe folgt ein Null-Byte, um Ihrem Amiga mitzuteilen, daß die Beschreibung dieses Funktionsaufrufes beendet ist. Für Funktionen, die sich an die Konventionen der Parameterübergabe der Sprache C halten, ist die Registerparameterliste leer, da die Parameter dann über dem Stack übergeben werden. Wenn Sie sich erst Ihr File mathtrans.bmap erzeugt haben, dann haben Sie auch eine wichtige Hürde auf dem Weg zur Benutzung der Library genommen.

Das FFP-Format

Bei einem Aufruf einer Library-Funktion müssen Sie genau unterscheiden, ob diese Funktion ein Resultat zurückgibt oder nicht. Falls keiner Ihrer Funktionsaufrufe einen Wert an Ihr Basic-Programm zurückgibt, genügt es, die Library mit dem Befehl LIBRARY Libraryname dem Basic-Interpreter bekannt zu machen. In unserem Beispiel (Listing 5) ist dies nicht der Fall und wir müssen Basic mitteilen, welchen Variablentyp die Funktionen zurückliefern. Nun ist eine FFP-Variable gerade so lang, wie eine lange Ganzzahl. Deshalb haben wir alle Funktionen mit Hilfe von

```
DECLARE FUNCTION Funktionsname LIBRARY
```

als solche definiert. Die in Klammern angegebenen Funktionsargumente müssen Sie nicht angeben, aber Sie bieten eine wertvolle Gedächtnisstütze.

Nach der Erzeugung der .bmap-Dateien treffen wir hier auf unser zweites Problem: Wie bringen wir unsere normalen Gleitpunktzahlen in das FFP-Format? Die Zuweisung »ffp& = Gleitzahl« löst unser Problem sicher nicht, da bei einer derartigen Zuweisung der Nachkommaanteil einfach abgeschnitten wird. Das liegt ja nun nicht in unserem Sinne. Auch die einfache Benutzung einer normalen Gleitpunktzahl ist nicht der richtige Weg, da sich das FFP-Format leider vom bei den normalen Gleitpunktzahlen verwendeten IEEE-Format unterscheidet. Wir kommen aber zum Ziel, wenn wir einen kleinen Umweg über Strings in Kauf nehmen. Mit Hilfe der Funktion MKS$ (Gleitpunktzahl) wird ein 4 Byte langer String produziert, aus dem mit der Funktion CVL (<String>) wieder eine lange Ganzzahl gewonnen werden kann. Nun können Sie diese Ganzzahl an die Funktion »SPFieee« übergeben, die daraus eine lange Ganzzahl macht. Da dies während solcher Problemstellungen öfter vorkommen kann, haben wir für Sie die Funktion »FNftoffp&« definiert, die eine Gleitzahl in das FFP-Format überführt. Die danach definierte Funktion »FNffptof« leistet genau das Umgekehrte: Sie wandelt eine FFP-Zahl in eine Gleitpunktzahl um.

In unserem Beispiel wird nun der IEEE-Ganzzahl »ieee« der Wert 0.5 zugewiesen, dieser Wert dann in das FFP-Format konvertiert und danach eine Reihe von transzendenten Funktionen aufgerufen und deren Ergebnis ausgegeben. Zu guter Letzt lassen sich mit dem Befehl

```
LIBRARY CLOSE
```

die geöffneten Libraries wieder schließen.

Mit dieser sicher nicht ganz leichten Thematik beschließen wir unseren Basic-Kurs. Wenn Sie unsere Reihe aufmerksam verfolgt haben, besitzen Sie jetzt nicht nur eine Menge neues Wissen, sondern auch ein Vielzahl von Anregungen für eigene Experimente. Setzen Sie Ihre Ideen in die Tat um. Bauen Sie das vorhandene Wissen durch »Learning by Doing« aus. Vielleicht steht dann eines Tages Ihr Name unter dem Programm des Monats im AMIGA-Magazin.

(Jürgen Singer/pa)

```
DEFINT a-Z : REM Alle Variablen Integer
READ CodeLength
DIM McProg(CodeLength)
FOR i = 0 TO CodeLength - 1
  READ McProg(i)
NEXT i
PRINT "Geben Sie einen String ein: "
INPUT st$
UPPER st$
PRINT st$
END

SUB UPPER(argument$) STATIC
  SHARED McProg()
  adr& = SADD(argument$)
  laeng& = LEN(argument$)
  entry& = VARPTR(McProg(0))
  CALL entry&(laeng&,adr&)
END SUB

DATA 27
DATA &h48e7,&hc080
DATA &h202f,&h0010
DATA &h206f,&h0014
DATA &h4281
DATA &h6000,&h001c
DATA &h1230,&h0000
DATA &h0c01,&h0061
DATA &h6d00,&h0010
DATA &h0c01,&h007a
DATA &h6e00,&h0008
DATA &h0230,&h00df,&h0000
DATA &h51c8,&hffe4
DATA &h4cdf,&h0103
DATA &h4e75
```

Listing 4. Maschinencode im Basic-Programm

```
DECLARE FUNCTION SPAsin& (ffp&) LIBRARY
DECLARE FUNCTION SPAcos& (ffp&) LIBRARY
DECLARE FUNCTION SPAtan& (ffp&) LIBRARY
DECLARE FUNCTION SPSin&  (ffp&) LIBRARY
DECLARE FUNCTION SPCos&  (ffp&) LIBRARY
DECLARE FUNCTION SPtan&  (ffp&) LIBRARY
DECLARE FUNCTION SPSinh& (ffp&) LIBRARY
DECLARE FUNCTION SPCosh& (ffp&) LIBRARY
DECLARE FUNCTION SPTanh& (ffp&) LIBRARY
DECLARE FUNCTION SPExp&  (ffp&) LIBRARY
DECLARE FUNCTION SPLog&  (ffp&) LIBRARY
DECLARE FUNCTION SPLog10&(ffp&) LIBRARY
DECLARE FUNCTION SPSqrt& (ffp&) LIBRARY
DECLARE FUNCTION SPTieee&(ffp&) LIBRARY
DECLARE FUNCTION SPFieee&(INT &O0) LIBRARY

LIBRARY "mathtrans.library"

DEF FNftoffp&(float) = SPFieee&(CVL(MKS$(float)))
DEF FNffptof(ffp&)   = CVS(MKL$(SPTieee&(ffp&)))

ieee       = .5
ffp&       = FNftoffp&(ieee)
ffpAsin&   = SPAsin&(ffp&)  : PRINT "arcsin(";ieee;") = ";
FNffptof(ffpAsin&)
ffpAcos&   = SPAcos&(ffp&)  : PRINT "arccos(";ieee;") = ";
FNffptof(ffpAcos&)
ffpAtan&   = SPAtan&(ffp&)  : PRINT "arctan(";ieee;") = ";
FNffptof(ffpAtan&)
ffpSin&    = SPSin&(ffp&)   : PRINT "   sin(";ieee;") = ";
FNffptof(ffpSin&)
ffpCos&    = SPCos&(ffp&)   : PRINT "   cos(";ieee;") = ";
FNffptof(ffpCos&)
ffpTan&    = SPtan&(ffp&)   : PRINT "   tan(";ieee;") = ";
FNffptof(ffpTan&)
ffpSinh&   = SPSinh&(ffp&)  : PRINT "  sinh(";ieee;") = ";
FNffptof(ffpSinh&)
ffpCosh&   = SPCosh&(ffp&)  : PRINT "  cosh(";ie-
ee;") = ";FNffptof(ffpCosh&)
ffpTanh&   = SPTanh&(ffp&)  : PRINT "  tanh(";ieee;") = ";
FNffptof(ffpTanh&)
ffpExp&    = SPExp&(ffp&)   : PRINT "   exp(";ieee;") = ";
FNffptof(ffpExp&)
ffpLog&    = SPLog&(ffp&)   : PRINT "   log(";ieee;") = ";
FNffptof(ffpLog&)
ffpLog10&  = SPLog10&(ffp&) : PRINT " log10(";ieee;") = ";
FNffptof(ffpLog10&)
ffpSqrt&   = SPSqrt&(ffp&)  : PRINT "  Sqrt(";ieee;") = ";
FNffptof(ffpSqrt&)
LIBRARY CLOSE
END
```

Listing 5. Transzendente FFP-Funktionen

Grafik auf dem Amiga

Mit der flexiblen Hardware Ihres Computers lassen sich fantastische Animationseffekte oder Bilder mit bis zu 4096 Farben erzeugen. Wir beschreiben Ihnen, wie man diese Hardware steuern muß, um die gewünschten Effekte zu erzielen.

Im ersten Teil dieses Kurses haben wir die Hardware-Komponenten des Amiga, den Weg, den ein Byte vom Speicher bis zum Monitor zurücklegt und die Arbeitsweise mit Bitplanes vorgestellt. In diesem Teil wollen wir uns mit den verschiedenen Auflösungsmodi der grafischen Darstellung beschäftigen.

Die Bildinformationen werden beim Amiga in sogenannten Bitplanes, auch Speicherlandkarten genannt, gespeichert. Durch die Kombination mehrerer Bitplanes lassen sich entsprechend viele verschiedene Farben darstellen. Erinnern Sie sich noch? Bei der Benutzung von fünf Bitplanes im normalauflösenden Modus ist die Darstellung von maximal 32 Farben gleichzeitig möglich. Wegen der durch die heutige Speichertechnologie begrenzten Zugriffszeit sind im hochauflösenden Modus nur 16 Farben darstellbar.

Wechselschalter im Chip

Außer den bereits bekannten Auflösungen (320 x 256 Bildpunkte im LoRes-Modus und 640 x 256 Bildpunkte im HiRes-Modus) bietet der Amiga noch andere Grafikauflösungen, die teilweise aus den beiden Grundmethoden abgeleitet sind. Um dem Amiga mitzuteilen, welche Auflösung verwendet werden soll, stehen dem Programmierer bestimmte Speicherstellen zur Verfügung (Tabelle). Da sich diese Speicherstellen innerhalb der Custom-Chips befinden, nennt man sie auch Register. In diesen Registern werden die Informationen über die Grafikauflösung, Anzahl der Bitplanes und Startadresse der Bitplanes abgelegt. Das Kontrollregister, welches für die Auflösung zuständig ist, trägt den Namen BPLCON0. Dieses Register hat 16 Speicherzellen (Bit 0 bis bis Bit 15). Eine solche Speicherzelle kann man sich wie einen Schalter vorstellen, deren Stellung die jeweilige Funktion ein- oder ausschaltet. Für das Einschalten des hochauflösenden Grafikmodus ist das Bit Nummer 15 (auch HIRES genannt) zuständig. Hat das Bit 15 den Wert 1, ist der HiRes-Modus eingeschaltet. Befindet sich dort eine 0, ist die normale LoRes-Auflösung aktiviert.

Um die Anzahl der zu verwendenden Bitplanes zu bestimmen, könnte man für jede der sechs maximal zur Verfügung stehenden Planes ein eigenes Registerbit verwenden. Dies wäre allerdings Verschwendung. Man verwendet statt dessen nur drei Speicherstellen, deren Kombination den gewünschten Wert ergibt. Diese Speicherstellen sind die Bits Nummer 12, 13 und 14 des BPLCON-Registers. Die drei Bits tragen die Namen BPU0, BPU1 und BPU2. Aus der letzten Folge wissen Sie, daß man mit 3 Bit acht verschiedene Kombinationen aus Nullen und Einsen bilden kann. Das erste und achte Bitmuster wird jedoch nicht benutzt, und so sind nur Werte von 1 bis 6 erlaubt.

Offset* hex	dez	Register	Basisadresse der Custom-Chips: ($DFF000) 14675968
($100)	256	BPLCON0	Bitplane-Kontrollregister 0
($102)	258	BPLCON1	Bitplane-Kontrollregister 1
($104)	260	BPLCON2	Bitplane-Kontrollregister 2
($096)	150	DMACON	DMA-Kontrollregister
($0E0)	224	BPL0PTL	Bitplanezeiger Plane 0 (Bit 16 bis 19)
($0E2)	226	BPL0PTH	Bitplanezeiger Plane 0 (Bit 0 bis 15)
...			
($0F4)	244	BPL6PTL	Bitplanezeiger Plane 6 (Bit 16 bis 19)
($0F6)	246	BPL6PTH	Bitplanezeiger Plane 6 (Bit 0 bis 15)
($180)	384	COLOR00	Farbregister 0
...			
($1BE)	446	COLOR31	Farbregister 31
($08E)	142	DIWSTRT	Display-Window-Start
($090)	144	DIWSTOP	Display-Window-Stop
($092)	146	DDFSTRT	Data-Fetch-Start
($094)	148	DDFSTOP	Data-Fetch-Stop

* Registeroffset + Basisadresse = Registeradresse

Tabelle. Registerinhalte steuern die Hardware im Amiga und bestimmen damit das Aussehen der Grafikdarstellung

Bilder im Interlace

Vielleicht haben Sie schon mal Bilder mit einer zwar hohen Auflösung, aber auch störendem Bildflimmern gesehen. Diese Auflösungsform heißt Interlace. Interlace bedeutet soviel wie überlagert oder verschränkt. Die Arbeitsweise von Interlace läßt sich gut am Bespiel eines Fernsehgerätes demonstrieren:

Das normale Fernsehbild wird nicht 50mal pro Sekunde neu gezeichnet, sondern in zwei Durchgängen — den sogenannten Halbbildern — erstellt. Zuerst zeichnet der Elektronenstrahl alle ungeraden Zeilen, dann alle geraden Zeilen des Bildes. Würden nämlich alle 625 Zeilen des Fernsehbildes 50mal in der Sekunde zeichnen, müßte der Elektronenstrahl 31 250mal in der Sekunde über die Zeilen gelenkt werden. Bei der Einführung unserer Fernsehnorm war man allerdings noch nicht in der Lage, eine derart hohe Zeilenabtastgeschwindigkeit zu erreichen.

Bei der Verringerung der Einzelbilder pro Sekunde würde das menschliche Auge die einzelnen Bilder getrennt erkennen, was ein starkes Flimmern des Fernsehbildes zur Folge hätte. Also verfiel man auf die folgende Lösung:

Die Darstellung von 50 Einzelbildern pro Sekunde bleibt erhalten, jedoch wird das Bild in zwei Phasen aufgebaut. Im ersten Durchlauf werden alle Zeilen mit ungeraden Nummern übertragen und im zweiten dann alle geraden. Die beiden Halbbilder zusammen ergeben dann das Vollbild. Die Anzahl der Vollbilder liegt dadurch nur noch bei 25 pro Sekunde. Die Zeilenfrequenz kann so von 31 250 Hz um die Hälfte auf 15 625 Hz verringert werden. Aus Kompatibilitätsgründen verwendet auch der Amiga diese Technik.

Die Entwickler des Amiga haben sich entschieden, beide Halbbilder absolut identisch aufzubauen. Es wird also zweimal dieselbe Information übertragen. Dadurch liegt die Bildfrequenz bei 50 Hz und die Zeilenfrequenz bei 31 250 Hz. Als Folge davon ist die Zeilenzahl der gleichzeitig darstellbaren, verschiedenen Zeilen auf die Hälfte der eines Fensehbildes (625 Zeilen) beschränkt. Wenn Sie sich Ihr Monitorbild einmal aus der Nähe betrachten, erkennen Sie deutlich die Zwischenräume zwischen den Zeilen. Die Halbbilder werden nicht mehr versetzt zueinander gezeichnet.

Diese Aufteilung eines Bildes in zwei Halbbilder bezeichnet man als Interlace. Auch der Amiga kann seine vertikale

Teil 2

KURSÜBERSICHT

In diesem Kurs stellen wir Ihnen die Arbeitsweise der für die Bilddarstellung wichtigen Chips des Amiga vor. In den einzelnen Folgen des Kurses finden Sie folgende Themen behandelt:

TEIL 1: (Einführung): Die Spezialchips im Amiga; Bilddarstellung durch Punkte; Arbeitsweise von Monitor und Fernseher; Farbbildung durch Bitplanes

TEIL 2: (Playfields): Die Playfield-Hardware; der Datenzugriff durch Denise; Dual-Playfields; Scrolling; Hold & Modify; Interlace

TEIL 3: (Sprites): Datenstruktur, Positionierung und Bewegung von Sprites; virtuelle Sprites; Sprite-Attach

TEIL 4: (Blitter/Copper): Arbeitsmodi des Blitter; Programmierung des Copper; die Kontrollhardware

Auflösung mit dieser Technik erhöhen. Der Nachteil dieses Modus ist ein stärkeres Bildflimmern. Dieses Flimmern entsteht, da nun nur noch 25 (Halb-)Bilder pro Sekunde dargestellt werden. Ihr Auge kann jedoch 25 Bilder bereits als Einzelbilder unterscheiden.

Der Schalter für den Interlace-Modus befindet sich ebenfalls in dem Register BPLCON0. Durch Setzen des Bits Nummer 2 wird der Interlace-Modus eingeschaltet. Die Bezeichnung für dieses Bit lautet sinngemäß LACE.

Mit den bisher erworbenen Kenntnissen ist es uns möglich, den Auflösungsmodus und die Anzahl der Farben unserer späteren Grafik zu bestimmen. Es ist jedoch noch ein weiter Weg bis zum Erscheinen der Grafik auf dem Bildschirm. Wie wir bereits in der letzten Folge erläutert haben, wird die Zahl der darstellbaren Farben durch die Anzahl der verwendeten Bitplanes bestimmt. Eine Gruppe zusammengehöriger Bit-Planes bezeichnet man als Playfield. Ein Playfield ist also eine Ansammlung von einer oder mehreren Bitplanes, die zusammen ein Grafikbild ergeben. Die bereits im vorigen Teil des Kurses beschriebenen Farbregister besitzen die Namen COLOR0, COLOR1,... bis COLOR31. Es sind, wie das BPLCON-Register auch, 16-Bit-Register, von denen jedoch nur 12 Bit — vier für jeden Farbanteil — genutzt werden.

Die Playfield-Hardware

Sind die Farben gesetzt, muß dem Computer die Anzahl und Größe der benötigten Bitplanes mitgeteilt werden. Im Speicher müssen die Daten für jede Bitplane an einem Stück abgelegt sein. Innerhalb der Grafikdaten dürfen sich also keine anderen Programmteile befinden. Die Angabe des reservierten Grafikspeichers nehmen wir durch das Setzen von jeweils zwei Adreßregistern für jede Bitplane vor. Die Register heißen BPLxPTH und BPLxPTL. Das x steht für die Bitplane 1 bis 6. Registerpaare mit den Endungen PTH und PTL geben immer 19-Bit-Adressen an. 19 Bit deshalb, weil die Customchips nur auf die unteren 512 KByte Zugriff haben.

Wir müssen nicht unbedingt die gesamte Grafik auf dem Bildschirm darstellen. Der Amiga kann einen kleineren Ausschnitt aus einem größeren Playfield (Bild 1) anzeigen. Diesen Teilausschnitt nennt man Display-Window. Die Größe des Display-Windows wird festgelegt, indem wir die horizontalen und vertikalen Start- und Endkoordinaten angeben. Die Startkoordinate bestimmt, wo das Display-Window auf dem Bildschirm beginnt; die Endkoordinaten legen das Ende des Windows und damit auch die Größe des Playfield-Ausschnittes fest. Obwohl das Window in der ersten vom Elektronenstrahl abgefahrenen Zeile beginnen könnte, ist die erste vertikale Position des Display-Windows in der Regel bei Zeile 44 und die erste horizontale Position auf Spalte 129 eingestellt (siehe Overscan im ersten Teil dieses Kurses).

Das Register, aus dem der Videochip die Information über die linke obere Ecke des Display-Windows gibt, heißt DIWSTART. Dieses Register enthält sowohl die x- als auch die y- Koordinaten. Die Koordinaten sind in einem Koordinatensystem angegeben, das dem LoRes-Non-Interlaced-Modus entspricht. Da das Register 16 Bit breit ist, benutzt es jeweils ein Byte (8 Bit) für die x- und ein Byte für die y-Koordinate. Diese Bytes nennen wir HSTART und VSTART. Da ein Byte höchstens einen Wert von 256 ergibt, können wir die x-Koordinate der Startposition nicht weiter als in das vierte Fünftel des Bildschirms versetzen. Da die y- Koordinate in der Regel nur bis 200 geht, wird uns durch die Bestimmung mit einem Byte keine Beschränkung auferlegt.

Grenzen der Darstellung

Haben wir die Startposition gesetzt, muß noch die rechte untere Ecke des Display-Windows festgelegt werden. Diese wird durch das Register DIWSTOP kontrolliert. Auch hier ist die Koordinatenangabe durch

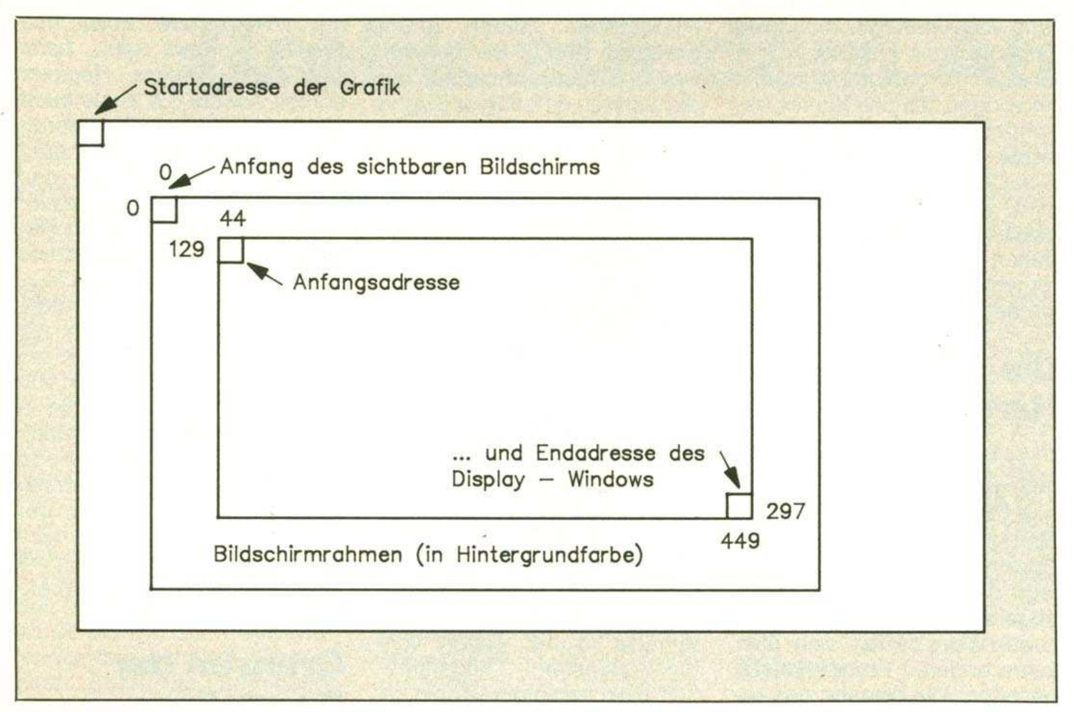


Bild 1. Das Display-Window läßt sich (fast) beliebig am Bildschirm positionieren

2 Byte, HSTOP und VSTOP, realisiert. Da wir allerdings höchstens einen Wert von 256 auf diese Weise darstellen können, addiert die Videologik automatisch den Wert 256 auf den Inhalt von HSTOP.

Der Greifer in Aktion

Das Aussehen des Bildes haben wir durch die Festlegung von Display-Window, Auflösung und Bitplanes bereits festgelegt. Jetzt müssen die Daten noch aus dem Speicher geholt und an der richtigen Stelle des Bildschirms angezeigt werden. Diesen Vorgang nennt man »Data Fetching«. Dazu wird die horizontale Zeilenposition, an der das Data Fetching beginnen beziehungsweise enden soll, in ein Register geschrieben. Die Data-Fetching-Register haben eine vier Punkte große Auflösung. Jede Position ist demnach vier Pixel von der letzten entfernt. Pixel 0 entspricht der Position 0; Pixel 4 der Position 1 und so weiter. Horizontales Scrolling und die Data-Fetch-Position sind voneinander abhängig. Es wird empfohlen, die Data-Fetch-Startwerte auf eine Auflösung von 16 Pixel zu beschränken. Die Hardware benötigt einige Zeit nach dem ersten Holen der Daten, bevor diese angezeigt werden können. Aus diesem Grund besteht ein Unterschied zwischen dem Wert des Display-Window-Starts und dem Data-Fetch-Startwert. Der normale Wert des DDFSTART-Registers im LoRes-Modus liegt bei 56 (hexadezimal: $38), im HiRes-Modus bei 60 ($3c).

Dem System muß außerdem mitgeteilt werden, welche Bytes in jeder Zeile angezeigt werden. Um das zu tun, muß der sogenannte Modulus festgelegt werden. Der Modulus ist die Zahl der Bytes im Speicher, die zwischen dem letzten Speicherwort einer Zeile und dem ersten Wort der nächsten Zeile liegen. So werden die in linearer Form (von Startadresse bis Endadresse) gespeicherten Bitplane-Daten in rechteckige Form (also eine Zeile sequentieller Daten, gefolgt von der nächsten Zeile) gebracht.

Für ein einfaches Playfield, bei dem die Bitplanegröße der Display-Window-Größe entspricht, ist der Modulus 0, denn am Ende des letzten Datenwortes einer Zeile folgen gleich ohne Zwischenschritte die Daten für die nächste Zeile (Bild 2). Ist das Playfield breiter als der Teil, der angezeigt werden soll, ist der Modulus gleich der Anzahl der Pixel des Playfields abzüglich der Anzahl der Playfieldpixel, die angezeigt werden sollen. Dadurch können wir Playfields darstellen, die größer sind als das Display-Window.

Die Bitplane-Adreßzeiger BPLxPTH und BPLxPTL werden vom System benutzt, um die Daten durch das Data-Fetching auf den Bildschirm zu bringen. Die Bitplane-Adreßzeiger sind dynamische Register, das heißt sie ändern sich ständig. Der Wert, den wir zu Beginn festgelegt haben, ist nur der Anfangswert. Beginnt der Data-Fetch-Vorgang, werden die Adreßzeiger fortlaufend um ein Speicherwort inkrementiert. Diesen Vorgang übernimmt das Betriebssystem des Amiga automatisch.

Im Interlace-Modus funktioniert der Data- Fetch etwas anders. Die Bitplanes für ein Interlace-Display sind 400 Zeilen lang (hoch) und nicht 200. Hat das Playfield im Speicher beispielsweise die normale Größe von 320 Pixel pro Zeile, dann beginnen die Daten für ein Interlace-Bild an den Byteadressen START (ein von uns festgelegter Wert) für die erste Zeile. Die zweite Zeile beginnt bei START+40, die dritte Zeile bei START+80 und so weiter. Wir benutzen deshalb einen Modulus von 40, um die Linien des anderen Feldes zu überspringen. Dadurch können wir eine Interlace-Grafik im Speicher so anlegen, als hätten wir eine vertikale Auflösung von 400 Punkten.

Nach dem Setzen der notwendigen Register starten wir die Playfield-Anzeige durch Anschalten der Bitplane-DMA. DMA bedeutet Direct Memory Access und ist die Fähigkeit der Custom-Chips direkt auf den Speicher zurückzugreifen, ohne den Prozessor dafür zu benutzen. Das Anschalten des direkten Speicherzugriffs für die Anzeige der Playfields erfolgt durch Setzen des Bit 8 (BPLEN: BitPlaneENable) im Register DMACON (DMA-Control). Erst dann wird die Darstellung der Grafik auf dem Bildschirm in Bewegung gesetzt: das Data-Fetching läuft.

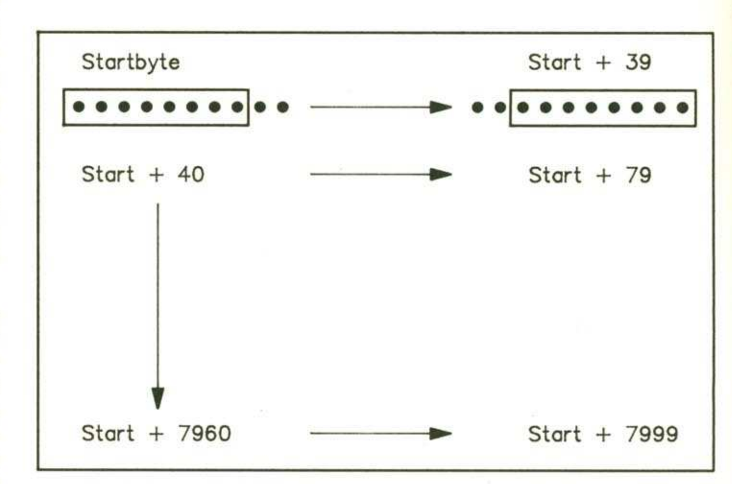


Bild 2. Anordnung der Bildinformationen im Speicher

Nach jedem Inkrementieren wird das aktuelle Wort aus dem Bitplane-Adreßregister geholt und angezeigt. Wenn die Bedingung des Zeilenendes erfüllt ist (bedingt durch DDFSTOP), wird der Modulus von der Hardware auf die Bitplanezeiger addiert, damit die nächste Zeile richtig angezeigt werden kann. Bei größeren Playfields wird also der Bereich zwischen DDFSTOP und dem durch den Modulus festgelegten neuen Zeilenanfang nicht mehr angezeigt.

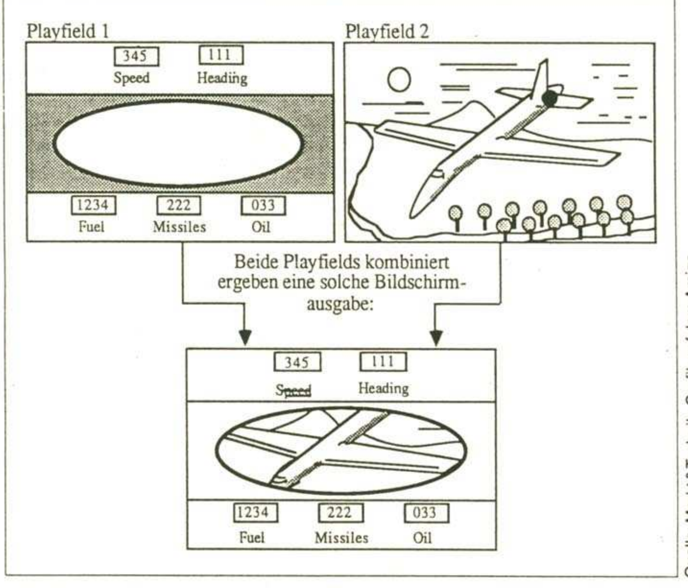
Direkter Speicherzugriff

Jedesmal, wenn das Playfield wieder angezeigt werden soll, müssen allerdings die Bitplanezeiger wieder auf den Anfang des Bildspeichers zurückgesetzt werden, da der Zeiger durch das Inkrementieren auf einen Speicherbereich zeigen

würde, in dem keine Grafikdaten mehr vorhanden sind. Ist das Ende des Playfields erreicht, setzen wir also die Bitplanezeiger wieder auf den Anfang der Grafik.

Dual-Playfields

Bei vielen Spielen haben Sie sicherlich schon einmal einen Effekt gesehen, bei dem es so aussah, als ob der Spieler aus einem Fenster auf einen bewegten Hintergrund blickt. Spiele, die eine Fahrzeugsimulation projizieren, sind ein gutes Beispiel hierfür. Im Vordergrund befinden sich alle Bedienungselemente des Fahrzeugs. Durch ein Fenster blickend sieht man auf eine bewegte Landschaft. Das Entstehen dieses Effekts kann man sich wie bei der Hinterglasmalerei vorstellen. Sie halten das Glasbild in der Hand, blicken durch die offenen, nicht bemalten Stellen des Glases auf den Hintergrund und bewegen das Bild. Mit etwas Fantasie haben Sie das Gefühl, durch ein Fenster auf einen bewegten Hintergrund zu blicken.



Quelle: Markt & Technik, Grafik auf dem Amiga

Bild 3. Die Kombination zweier Playfields zu einem Bild erleichtert die Programmierung bewegter Landschaften

Um diesen Effekt am Amiga zu erzeugen, benötigen wir zwei Bilder, die sich völlig getrennt voneinander bewegen lassen. Ein Playfield wird vor dem anderen gezeigt, wobei Teile des einen Playfields durchsichtig sein müssen, damit man auf das andere hindurchsehen kann. Es handelt sich hierbei um eine Dual-Playfield-Grafik (Bild 3). Um den Dual-Playfield-Modus zu aktivieren, setzen wir in unserem altbekannten BPLCON0-Register das Bit Nummer 10 auf 1. Dieses Bit trägt sinngemäß den Namen DBPLF.

Wie wir inzwischen wissen, kann der Amiga maximal sechs Bitplanes verwalten. Setzen wir das DBPLF-Bit, so werden die maximal möglichen Bitplanes durch zwei geteilt. Für jedes Playfield stehen damit maximal drei Bitplanes zur Verfügung. Dies entspricht einer Auflösung von acht Farben. Jedes Playfield kann so seine eigene Farbpalette erhalten. Die Hardware faßt die Bitplanes mit ungeraden Nummern (1, 3 und 5) zum Playfield 1 und die anderen (Nummer 2,4 und 6) zum Playfield 2 zusammen (HiRes-Modus: nur 2 Planes pro Playfield möglich).

Das Playfield im Vordergrund

Playfield 1 nutzt die Farbregister 0 bis 7 und Playfield 2 die restlichen Register 8 bis 15.

Welches Playfield im Vordergrund oder Hintergrund gezeigt wird, entscheidet das sogenannte PF2PRI-(PlayField-2-PRiority)Bit im BPLCON2-Register. Ist dieses Kontrollbit

gesetzt, wird Playfield 2 im Vordergrund und Playfield 1 als Hintergrund angezeigt. Der umgekehrte Zustand ergibt sich bei gelöschtem PF2PRI. Wo immer die Farbe (das Register) 0 (transparent) angewählt wurde, erscheint die Farbe desjenigen Objektes, das hinter dem Playfield liegt.

Außer der Darstellung von Playfields, die der Größe der Display-Windows entsprechen, haben wir noch die Möglichkeit, größere oder kleinere Grafiken darzustellen. Kleinere Grafiken auf den Bildschirm zu bringen, dürfte uns mit den jetzigen Kenntnissen nicht schwer fallen. Mit Hilfe der Data-Fetch Registerinhalte passen wir das Display-Window unseren Wünschen an. Ist ein Bild größer als das Display-Window, so müssen wir definieren, welcher Teil der Grafik angezeigt werden soll. Die Grafik wird im Speicher einfach so angelegt, als ob wir eine Bildschirmauflösung hätten, die der Größe des Playfields entspricht. Als Beispiel nehmen wir eine Grafik mit 640 Bildpunkten pro Zeile, also eine Grafik, die doppelt so breit ist wie normal. Der Amiga würde nach dem 40. Byte die Zeile wechseln und das 41. Byte in der zweiten Zeile auf den Bildschirm bringen. In Wirklichkeit wollten wir jedoch erst nach dem achtzigsten Byte die Zeile wechseln und Byte 41 bis 80 sollten unterdrückt und nicht zur Anzeige gebracht werden.

Playfield-Scrolling

Wie wir wissen, wird der Rasterstrahl nach jedem Zeilendurchlauf dunkelgetastet und an den Anfang der nächsten Zeile zurückgesetzt. Diesen Vorgang nennt man vertikal Blanking. Beim Amiga besteht nun die Möglichkeit, in dieser kurzen Pause mit Hilfe des Coppers in eine Vertical Blanking Interrupt-Routine zu verzweigen. Nach jedem Rasterstrahldurchlauf erfolgt automatisch ein sogenannter Interrupt. Dies ist ein hardwaremäßig erzeugtes Unterbrechungssignal, das den Amiga zur Abarbeitung einer bestimmten Befehlsliste, die Copperliste genannt wird, zwingt. Im Normalfall befinden sich in dieser Befehlsliste Anweisungen, die das Zurücksetzen der Bitplane-Adreßzeiger auf den Anfang des Bildes bewirken. Wollen wir zum Beispiel nur die rechte Hälfte des Bildes anzeigen, ändern wir einfach die Interruptroutine, um in die Bitplane-Adreßzeiger den Wert Start+40 als neuen Startwert zu schreiben. Der Modulus bleibt bei 40, da auch hier 40 Byte übersprungen werden müssen. In unserem Fall sind es die 40 Byte, die links liegen und verschwinden sollen. Am Ende der Zeile angekommen, enthalten die Bitplanezeiger den Wert START+80. Hierzu wird der Modulus von 40 dazuaddiert und wir erhalten den Wert START+120. Damit befinden wir uns wieder in der rechten Hälfte des Bildes. Durch Ändern der entsprechenden Werte können wir auch jeden anderen Teil des Bildes anzeigen. Wollen wir beispielsweise eine Grafik anzeigen, die höher ist als unser Display-Window, so addieren wir als Startwert bei einem LoRes-Bild entsprechend der zu überspringenden Zeilen jeweils den Wert 40 hinzu. (Wir werden in der vierten Folge dieses Kurses noch ausführlich auf die Nutzung des Coppers bei der Programmierung grafischer Effekte eingehen.)

Sicherlich haben Sie schon öfter Grafiken mit bewegtem Hintergrund bewundert. Vertikales Scrolling ist einfach zu bewerkstelligen. Bei jedem vertikalen Blanking-Vorgang ändern wir einfach die Bitplane-Adreßzeiger auf die nachfolgende oder vorhergehende Zeile der Grafik, je nach Richtung des Scrolling. Um beim horizontalen Scrolling von links nach rechts oder von rechts nach links zu scrollen, muß der Data-Fetch-Vorgang um ein Wort (16 Bit) erweitert werden. Das Anzeigen dieses Wortes muß dann entsprechend der Scroll-Geschwindigkeit einer Verzögerung unterliegen. Beide Typen des Scrolling haben aber gemeinsam, daß die Bitplane-Adreßzeiger während des vertikalen Blankings geändert werden. Angenommen, wir wollen ein langes Playfield von unten nach oben scrollen. Während des Vertikal-Blanking zählen wir dann zum ursprünglichen Wert START, bei dem das Data-Fetching beginnt, den Bytewert, der der Zahl der Pixel pro Zeile entspricht, dazu. Um ein Playfield von links nach rechts zu scrollen, muß die Geschwindigkeit des Scrollings durch die Angabe des Fetching in Pixel kontrolliert werden. Diese Verzögerung bedeutet, daß ein zusätzliches Datenwort vom Speicher auf den Bildschirm gebracht, aber nicht sofort angezeigt wird. Scrollt man nach rechts, wird das zusätzliche Wort links vom sichtbaren Teil des Bildes kurz vor dem DDFSTART-Wert plaziert. Wenn das Display sich nach rechts bewegt, erscheinen die zusätzlichen Daten des linken Wortes auf der linken Seite des Display-Windows, während die Daten des rechtsliegenden Wortes nach rechts verschwinden. Je größer der Wert DELAY gewählt wird, desto größer ist die Geschwindigkeit des Scrollings.

Es ist ein DELAY-Wert bis zu 15 Pixel möglich. Die Verzögerung wird angeschaltet durch das Setzen der DELAY-Bits im BPLCON1-Register. Es handelt sich dabei um die Bits 0 bis 7. Die Bits 0 bis 3 geben den Wert PF1H (Playfield 1 Horizontal) für die Bitplanes ungerader Zahl ab, Bit 4 bis 7 ergeben den Wert PF2H für die Verzögerung der Bitplanes gerader Zahl. Bei Single-Playfield-Betrieb müssen PF1H und PF2H simultan auf den gewünschten Wert gesetzt werden, im Dual-Playfield-Modus müssen die Werte getrennt für die einzelnen Playfields gesetzt werden. Die Animation von Playfields ist auch mit Hilfe von Bibliotheksfunktionen des Betriebssystems möglich. Diese sind etwas einfacher zu benutzen als das direkte Ansprechen der Hardware.

Durch die Verwendung von sechs Bitplanes im niedrigauflösendem Modus könnten 64 Farben gleichzeitig dargestellt werden. Da aber nur 32 Farbregister vorhanden sind, kann der Amiga eigentlich nur 32 Farben gleichzeitig verwalten.

Mehr Farbe mit Extra-Halfbright

Dafür genügen fünf Bitplanes. Durch einen Trick kann man diese Beschränkung umgehen. Die Bits 0 bis 4 aus den Bitplanes 1 bis 5 dienen weiterhin als ein Zeiger auf ein Farbregister. Bit 5 aus der Bitplane 6 steuert den Extra-Halfbright-Modus. Ist der Wert in dieser Bitplane = 0, wird der Inhalt der Farbregister direkt auf dem Bildschirm ausgegeben. Liegt dieses Bit aber auf 1, wird der Farbwert vor der Anzeige durch zwei dividiert. Danach sind die einzelnen Farbwerte genau halb so groß. Wir haben es zwar immer noch mit nur 32 Farben zu tun, können aber durch die Verwendung der zwei Helligkeitsstufen gute Farbübergänge und Schattierungen erzielen. Angewählt wird der Extra-Halfbright-Modus durch die Verwendung von allen sechs Bitplanes.

Der Hold & Modify-Modus

Die Darstellung von 4096 Farben gleichzeitig auf dem Bildschirm läßt sich nur mit Einschränkungen praktizieren. Obwohl der Hold & Modify-Modus (H.A.M.-Modus) fester Bestandteil der Playfield-Hardware ist, kann er mit dem Aufbau der bisherigen Playfields nicht verglichen werden. Im H.A.M.-Modus werden die Farbwerte für einen Punkt vom vorherigen Punkt übernommen und eventuell geringfügig verändert. Dadurch lassen sich feine Farbabstufungen erzeugen. Die Methode wird durch ihren Namen sehr treffend beschrieben, Hold für das Halten des Farbwertes, Modify für die Änderung der Farbkomponente. Durch Setzen des Bit 11 (HOMOD) im BPLCON0-Register wird der HAM-Modus aktiviert. Die Bitplanes 5 und 6 steuern die Modifikationen. Ist die Kombination der Bits 5 und 6 = 00, so nimmt die Hardware eine normale Farbwahl vor. Ist die Kombination aus Bit 5 und 6 = 01, wird die Farbe des unmittelbar links liegenden Punktes gehalten und der Blauwert den Werten der Planes 1 bis 4 entnommen. Kombinationen von 10 und 11 bewirken die Modifikation der Farben Rot und Grün. Auf diese Weise können alle 4096 Farben, zu denen der Amiga überhaupt fähig ist, auf dem Bildschirm dargestellt werden. Die beschriebene Technik verhindert jedoch die freie Wahl einer Farbe für jeden beliebigen Punkt. Dies ist erst spätestens nach drei Punkten möglich, indem man nacheinander alle drei Komponenten der Pixelfarbe austauscht. Die 16 Farben, die sich über die Farbregister direkt anwählen lassen, können natürlich auch noch eingesetzt werden. Es gibt inzwischen schon Peripheriegeräte auf dem Markt, mit denen Sie von einer Videokamera aufgenommene Bilder im H.A.M.-Modus auf dem Amiga darstellen können.

Damit sind wir am Ende dieses Kursteiles. In der nächsten Folge beschäftigen wir uns eingehend mit der Programmierung von Sprites.

(Joschy Polierer/pa)

CLI – Die zweite Haut des Amiga

Nachdem im letzten Kursteil Befehle zur Programmierung einer Batch-Datei (Stapeldatei) behandelt wurden, erklären wir diesmal Spezialbefehle für Zusatzgeräte, die sich teilweise auch zur Einbindung in eine ausführbare Stapeldatei eignen.

Das CLI (Command-Line-Interface) ist ein mächtiges Anwender-Werkzeug im Umgang mit dem Amiga. Über das CLI sind viele Dateioperationen möglich, die über die Workbench-Ebene nicht zu realisieren sind. In diesem Teil des Kurses gehen wir unter anderem auf die Befehle ein, die etwas mit Zusatzgeräten wie beispielsweise Festplatten oder Zusatzlaufwerken zu tun haben.

Der erste dieser Spezialbefehle ist DJMOUNT. Dieser Befehl kann nur dann eingesetzt werden, wenn Sie einen Amiga 2000 mit PC-/AT-Karte besitzen, in den eine Festplatte eingebaut ist, die über einen Controller auf der MS-DOS-Seite gesteuert wird. Dieses Kommando sagt dem Amiga, daß er die für ihn eingerichtete Partitionierung auf der PC-Festplatte benutzen soll. Dazu eine kurze Erklärung: Bei der Partitionierung wird eine Festplatte in einzelne Bereiche aufgeteilt.

Bereiche werden reserviert

Jeder Bereich kann von einem anderen Betriebssystem belegt werden. Angenommen, Sie besitzen eine gängige 20-MByte-Festplatte mit zwei Platten und vier Köpfen, wobei jede Platte 612 Spuren hat (0 bis 611). Es wird gewünscht, daß jeweils 10 MByte für MS-DOS und 10 MByte für Amiga-DOS zur Verfügung stehen. Also wird der Festplatte (genauer: dem Festplatten-Controller) mitgeteilt, daß die Spuren 0 bis 305 für MS-DOS und die Spuren 306 bis 611 für den Amiga reserviert sind. Die Platte besitzt also zwei Partitions (Bild).

Fremde Partitions werden nicht erkannt. Das bedeutet, daß für den Amiga die ganze Platte nur 306 Spuren groß ist, genauso, wie der PC/AT auch nur 306 Spuren der Platte für sich erkennt.

Der Befehl DJMOUNT darf aber erst aufgerufen werden, nachdem der PC/AT fertig gebootet hat und keine Plattenzugriffe innerhalb von AUTOEXEC.BAT mehr versucht. Gegebenenfalls muß mit WAIT genügend lange gewartet werden (etwa WAIT 40), da sonst das System abstürzen kann. Dies ist logisch, da nicht zwei Systeme gleichzeitig auf die Platte zugreifen können.

Nachdem der Befehl aufgerufen und ausgeführt wurde, steht dem A2000-Besitzer (mit Brückenkarte und PC-Festplatte) der für Amiga-DOS reservierte (partitionierte) Bereich zur Verfügung. In einer Startdatei könnte das so aussehen:

```
BINDDRIVERS
WAIT 40
DJMOUNT
ASSIGN devs: jh0:devs
ASSIGN sys: jh0:
ASSIGN c: jh0:c
...
...
```

In dieser Befehlsfolge wird zuerst der MS-DOS-Teil aktiviert. Der Amiga wartet dann so lange, bis der PC fertig gebootet hat und teilt Amiga-DOS anschließend mit, daß eine Festplatte (jh0:) zur Verfügung steht. Daraufhin werden die Standard-Dateiverzeichnisse so installiert, daß der Amiga von nun an alle Befehle, Fonts, Devices oder Libraries auf der Festplatte sucht.

Doch um die PC-Platte für den Amiga zugänglich zu machen, muß zuerst die Partition eingerichtet und die Platte formatiert werden. Dazu existieren die Befehle FDISK, ADISK (beide zu finden auf der MS-DOS-Systemdiskette) und DPFORMAT. Das Einrichten geschieht folgendermaßen:

— Zuerst mit BINDDRIVERS den PC/AT booten. Dann mit FDISK und ADISK die einzelnen Bereiche der Platte einrichten (beispielsweise 10 MByte für den PC und den Rest für den Amiga).

— Anschließend vom CLI aus DJMOUNT eingeben. Die auftretende Fehlermeldung kann ignoriert werden.

Um die Amiga-Partition zu formatieren, ist folgendes einzutippen:

```
DPFORMAT DRIVE jh0:
NAME Name
```

Der Amiga formatiert und validiert seinen Bereich, der im Anschluß vom Amiga aus als »jhx:« benutzt werden kann.

Sollten Sie aber einen SCSI-Controller (etwa den von Commodore) mit der dazugehörigen Festplatte besitzen, muß zum Formatieren der Hard-Disk ein anderer Befehl verwendet werden: PREP. Nachdem die Platte mit MOUNT (näheres dazu später) aktiviert wurde, muß eine Routine aufgerufen werden, die dem Festplatten-Controller die Daten der Platte übermittelt.

Auch für SCSI ist gesorgt

Dies erledigt PREP. Für die erste Hard-Disk sieht der Aufruf folgendermaßen aus:

```
PREP Laufwerk:
```

Es erscheinen der Reihe nach verschiedene Abfragen, die die Platte betreffen. Zuerst will PREP den Festplattentyp wissen. Für eine gebräuchliche 20-MByte-ST506-Festplatte mit zwei Platten und vier Köpfen wählt man am besten die Seagate ST225. Bei der folgenden Abfrage möchte der Amiga wissen, ob die Platte automatisch nach einigen Sekunden Inaktivität die Köpfe parken soll. Die Antwort hängt von Ihrem Geschmack ab.

Sollten Sie die Hard-Disk partitionieren wollen (geschieht in einem Eintrag in der MountList), ist anschließend anzugeben, bis wohin die erste Partition reichen soll. Die nächste Frage bezieht sich darauf, wieviel Pufferspeicher der Platte zugestanden wird (näheres darüber später bei ADDBUFFERS).

Bevor die Vorbereitungen abgeschlossen werden, will Amiga-DOS noch wissen, ob defekte Sektoren der Hard-Disk markiert werden sollen. Nach Beantwortung dieser

Teil 6

KURSÜBERSICHT

Der CLI-Kurs erläutert Ihnen in mehreren Folgen den Umgang mit der Amiga-Benutzerschnittstelle CLI (Command-Line-Interface). Am Ende haben Sie alle wichtigen Befehle kennengelernt und können die Kommandos nach eigenen Wünschen einsetzen. Die folgende Kursübersicht zeigt Ihnen, in welchen Kursteilen welche CLI-Kommandos erklärt werden:

TEIL 1: behandelte Befehle zum Anzeigen des Directories und Stellen der Uhr (DIR, LIST, CD, DATE, ENDCLI)

TEIL 2: erklärte Ihnen, wie Sie Dateien kopieren, löschen, umbenennen und Disketten formatieren können (FILENOTE, DELETE, RELABEL, RENAME, TYPE, COPY, DISKCOPY, INFO, FORMAT, INSTALL)

TEIL 3: zeigte unter anderem auf, wie Texte und Startdateien erstellt und bearbeitet werden (MAKEDIR, ED, PROMPT, WAIT, LOADWB, SETCLOCK, EXECUTE)

TEIL 4: befaßte sich mit Befehlen, die sich zum Einbau in eine Batch-Datei anbieten (ASSIGN, ECHO, FAILAT, IF, ENDIF, SAY, STACK, PROTECT, BINDDRIVERS)

TEIL 5: setzte die Erklärung von Kommandos für eine Batch-Datei fort (RUN, EDIT, JOIN, LAB, QUIT, SKIP, ADDMEM, BREAK)

TEIL 6: geht auf neue Spezialbefehle für Zusatzgeräte wie etwa Festplatten ein (SEARCH, NEWCLI, SORT, WHY, DJMOUNT, PREP, DPFORMAT, MOUNT, ADDBUFFERS, DISKCHANGE)

TEIL 7: schließt mit den letzten Workbench-Befehlen ab (CHANGETASKPRI, DISKDOCTOR, FAULT, SETDATE, SETPREFS, STATUS)

Frage wird die Platte zur Formatierung vorbereitet.

Formatiert wird die Hard-Disk anschließend wie gewohnt mit dem bekannten CLI-FORMAT-Befehl.

Wie bereits erwähnt, muß die an SCSI-Controller hängende Platte in das System eingebunden werden. Dies erledigt MOUNT. Insofern in der MountList (nähere Informationen darüber erhalten Sie im neuen Amiga-DOS-Handbuch und im Manual des SCSI-Controllers) die Platte eingetragen ist, genügt der Befehl

```
MOUNT dh0:
```

oder richtiger

```
MOUNT res0:
```

damit die Hard-Disk sofort dem Amiga als »dh0:« zur Verfügung steht.

Ebenfalls kurz erwähnt wurde ADDBUFFERS. Dieser hilfreiche Befehl bringt dem Anwender große Vorteile:

Normalerweise reserviert Amiga-DOS beim Einschalten oder einem Reset automatisch etwa 17 KByte Speicher für jedes angeschlossene Laufwerk (Sektor-Cache-Speicher). Vergrößert der Anwender nun diesen Pufferspeicher, können in diesem Puffer einige Daten der Floppystation mehr stehen, es muß also nicht so oft auf die Diskette zugegriffen werden, der Datentransport wird etwas schneller. Um das zu erreichen, muß ADDBUFFERS zusammen mit einem Parameter eingegeben werden, der die Größe des zu reservierenden Puffers angibt:

```
ADDBUFFERS df1: 40
```

reserviert 40 zusätzliche Pufferspeicher (jeder Pufferspeicher hat etwa eine Größe von 500 Byte). Größer sollte der Pufferspeicher aber nicht gemacht werden, da die Zugriffszeitvorteile ab einer bestimmten Puffergröße nicht mehr so groß sind und unnötig Speicher verbraucht wird. Theoretisch sind aber keine Grenzen gesetzt.

Ein 5¼-Zoll-Laufwerk wird erkannt

Besitzen Sie ein 5¼-Zoll-Diskettenlaufwerk, das nicht automatisch einen Diskettenwechsel erkennt? Dann hilft DISKCHANGE weiter. Mit diesem Befehl teilen Sie Amiga-DOS mit, daß im 5¼-Zoll-Laufwerk die Diskette gewechselt wurde. Angenommen, das Laufwerk ist »df2:«, dann lautet der komplette Befehl:

```
DISKCHANGE df2:
```

Wenn im Amiga-Modus nicht mit 5¼-Zoll-Diskettenlaufwerken gearbeitet wird, kann dieser Befehl getrost gelöscht oder vergessen werden.

Sie geben einen Befehl ein, der mit einer lapidaren Fehlermeldung abgebrochen wird. Meist sind aber diese Fehlermeldungen nicht gerade aussagekräftig. Wenn Sie etwas mehr darüber erfahren möchten, geben Sie WHY ein. Der Amiga sagt Ihnen dann etwas detaillierter, warum die Ausführung des Befehls fehlgeschlagen ist.

Möchten Sie gerne wissen, ob eine bestimmte Zeichenkette in einer Datei enthalten ist, oder wissen Sie nicht mehr, wie oft in einer Datei ein Begriff vorkommt? Dann hilft SEARCH weiter. Der Befehl durchsucht das angegebene Verzeichnis beziehungsweise die angegebene Datei nach einer bestimmten, vorgegebenen Zeichenkette. Als Beispiel soll die Startdatei im aktuellen Laufwerk nach dem Befehl RUN durchsucht werden:

```
SEARCH startup-sequence
 RUN
```

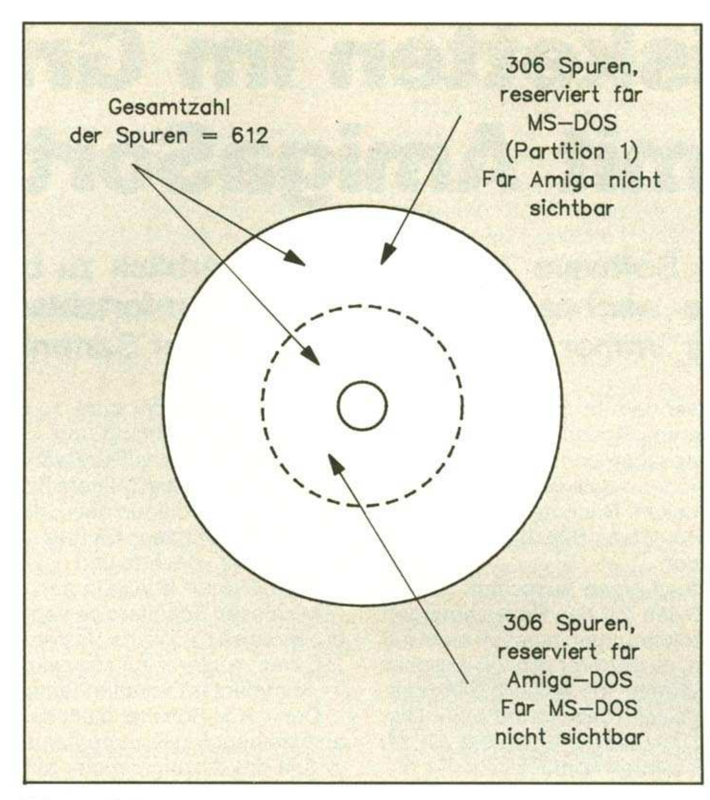
SEARCH durchsucht nun die Startdatei und zeigt alle Zeilen mit Zeilennummern und deren Inhalt an, in denen der Befehl RUN vorkommt. Wenn Sie nicht genau wissen, in welcher Datei der Suchtext steht, geben Sie einfach als Zusatzparameter ALL an. Suchen wir doch im gedachten Verzeichnis »Texte« nach dem Namen Müller:

```
SEARCH Texte Müller ALL
```

In den Suchstring können auch Joker, Platzhalter und Wildcards eingesetzt werden. Drei Beispiele:

```
SEARCH startup-sequence
 r#?
SEARCH df1: ma??r ALL
SEARCH jh0:texte/heu#?
```

Zu beachten ist, daß SEARCH zwischen Groß- und Kleinschreibung unterscheidet.



Dieses Bild verdeutlicht, wie eine Festplatte in Partitions, das heißt einzelne Teilbereiche, aufgeteilt ist

Warum kommt da ein Fehler?

Nützlich ist auch SORT. Angenommen, Sie möchten eine ASCII-Datei, die vielleicht Namen oder ähnliches enthält, alphabetisch sortieren. Dazu können Sie natürlich auch den ED aufrufen und mit seinen Hilfsfunktionen umständlich den Text neu sortieren. Oder Sie verwenden SORT. Der Befehl sortiert eine Textdatei in aufsteigender alphabetischer Ordnung. Dabei werden nur komplette Zeilen, die mit Return abgeschlossen sind, berücksichtigt. Außerdem muß die komplette Datei auf einmal in den Speicher passen. SORT unterscheidet nicht zwischen Groß- und Kleinschreibung.

Einen Haken hat SORT: Es muß genügend Stapelspeicher vorhanden sein, in dem SORT Daten zwischenlagern kann. Oftmals reichen aber die 4000 Byte, die normalerweise jedem CLI-Befehl zur Verfügung stehen, nicht aus. Damit der Amiga beim Sortieren nicht abstürzt, ist es zweckmäßig, den Stapelspeicher mit dem STACK-Befehl auf etwa 10 000 bis 15 000 Byte zu vergrößern.

Beim SORT-Befehl ist anzugeben, welche Datei sortiert werden (Quelldatei) und wie die sortierte Datei heißen soll (Zieldatei). SORT läßt die alte Datei unberührt und erstellt eine neue, sortierte Datei. Ein Beispiel: Wir möchten die Textdatei »Test« alphabetisch in aufsteigender Ordnung sortieren. Die neue Datei soll den Namen »Test.sort« erhalten:

```
SORT test TO test.sort
```

Wenn nicht gewünscht wird, daß ab der ersten Spalte einer jeden Zeile sortiert wird, läßt sich ein Zusatzparameter angeben, der festlegt, in welcher Spalte das erste zu sortierende Zeichen steht:

```
SORT test TO test.sort
COLSTART 8
```

Nach dieser Änderung wird erst ab dem achten Buchstaben jeder mit Return abgeschlossenen Zeile sortiert.

Zum Abschluß des Kursteils wieder ein kurzer, leicht verständlicher Befehl, der äußerst hilfreich ist. Es handelt sich dabei um NEWCLI. Angenommen, Sie möchten parallel zwei verschiedene CLI-Tätigkeiten ausführen oder benötigen mehrere offene CLI-Fenster (maximal 20). Geben Sie NEWCLI ein. Sofort besitzen Sie ein zweites CLI-Window, in dem genau wie im ersten gearbeitet werden kann. So läßt sich beispielsweise ein Text in einem Window ausgeben, während im anderen ein ähnlicher Text steht. Die zwei Texte lassen sich direkt vergleichen. Ein größerer Vorteil ist es aber, daß, falls ein Programm, das vom ersten CLI aus aufgerufen wurde, abstürzt, jederzeit ein weiteres CLI offen ist, in dem weitergearbeitet werden kann. Wie auch immer Sie die Möglichkeit, mehrere CLI-Fenster offen haben zu können, einsetzen, nützlich kann es sicher von Zeit zu Zeit sein. (dm)

Unterstützende Literatur:
Das Amiga-Handbuch, Markus Breuer, Markt & Technik-Verlag, 49 Mark

Disketten im Griff mit AmigaSort

Es gibt immer mehr Software für den Amiga und durch die wachsende Zahl von Disketten fällt es immer schwerer, den Überblick zu behalten. »AmigaSort« ist ein komfortables Werkzeug, um Ordnung in der Sammlung zu halten.

Viele Computerbesitzer verwenden ihr Gerät zur Verwaltung von Daten. Das Programm »AmigaSort« dient dazu, Diskettensammlungen zu verwalten und verfügt unter anderem über Sortier- und Suchfunktionen sowie über die Möglichkeit, übersichtliche Listen auszudrucken. Nach dem Starten von AmigaSort sieht man den Bildschirm (siehe Bild 1) vor sich, auf dem alle Aktionen ausgeführt werden.

Das große Fenster in der Mitte zeigt einen Ausschnitt der aktuellen Liste, bestehend aus fünf Zeilen mit den Bezeichnungen der jeweiligen Disketten. Diese Bezeichnungen müssen nicht mit dem Diskettennamen identisch sein, da letzterer sehr oft jeglicher Aussagekraft entbehrt, sie sind vielmehr frei editierbar. AmigaSort verwaltet also keine Inhaltsverzeichnisse, sondern nur Diskettennamen und dazugehörige Kommentare, in die die einzelnen Programmnamen eingetragen werden können.

Das kleine, langgezogene Fenster über dem Anzeigefenster zeigt den Diskettennamen zum aktuellen Eintrag, der im Anzeigefenster in der Mitte, das heißt in der dritten Zeile steht.

Das kleine Fenster rechts oben zeigt an, ob der aktuelle Eintrag einen Kommentar besitzt. Ist dies der Fall, so wird das durch ein großes »K« verdeutlicht.

In dem hohen, schmalen Fenster rechts neben dem Anzeigefenster erscheinen die Nummern der sichtbaren Einträge. Diese Nummern dienen ebenfalls nur dem Benutzer und sind wie die Diskettennamen frei editierbar.

In dem linken, kleinen Fenster unter dem Bedienungsfeld wird das aktuelle Laufwerk angezeigt, entweder »df0:« oder »df1:«.

Links daneben befindet sich der Status der Tasten. Da sämtliche Funktionen über die Tastatur aufrufbar sein sollten, es aber nur zehn Funktionstasten gibt, wurde ein Kompromiß geschlossen: Sämtliche Funktionen des Bedienungsfeldes werden über sinnbezogene Tastaturentsprechungen abgerufen, während alle Menüfunktionen über die Funktionstasten beziehungsweise die Escape-Taste gesteuert werden. Da dazu die Anzahl der Funktionstasten nicht ausreichen würde, kann mit der Help-Taste sozusagen zwischen »geshiftetem« und »ungeshiftetem« Modus umgeschaltet werden, was die Anzahl der Funktionstasten effektiv verdoppelt. Das Fenster ist blau bei ungeshiftetem Modus und weiß bei geshiftetem. Übrigens: Nachdem eine Funktion über Tastatur aufgerufen wurde, wechselt das Programm wieder in den ungeshifteten Modus, wodurch sich der Benutzer die Tastaturbelegung besser merken kann. Außerdem sind die ungeshifteten Tasten die meist gebrauchten.

Jeder Schalter des grafischen Bedienungsfeldes kann durch einen einfachen Mausklick mit der linken Maustaste betätigt werden. Es gibt acht Schalter:

— Anfang/Ende (Pfeile mit Querstrichen)

Das linke Tastenpaar dient dazu, an den Anfang beziehungsweise ans Ende der Liste zu gelangen. Die Tastaturäquivalente sind <—> für Anfang und <+> für Ende.

— Schnell Vorwärts/Rückwärts (Pfeile im schwarzen Feld)

Mit diesen Tasten blättern Sie in der Liste fünf Einträge vor- oder rückwärts. Sie dienen dazu, die Liste schnell auf der Suche nach einem bestimmten Eintrag durchzusehen. Tasten: <Cursor rechts> für vorwärts und <Cursor links> für rückwärts.

— Rauf/Runter (Pfeile in den orangen Feldern)

Mit diesen Schaltern bewegt man sich einen Eintrag vor- beziehungsweise rückwärts. Tasten: <Cursor hoch> für vorwärts und <Cursor runter> für rückwärts.

— Kommentar/Diskettenname (weißes »K« oder »D«)

Der <K>-Schalter dient dazu, den Kommentar eines Eintrags anzuzeigen. Ein Kommentar ist immer fünf Zeilen lang, das heißt er füllt das Anzeigefenster aus. Das Vorhandensein eines Kommentars wird in dem kleinen Fenster links oben durch ein »K« angezeigt.

Durch Drücken des Schalters <D> kann der Diskettenname der momentan im aktuellen Laufwerk befindlichen Diskette angezeigt werden.

Und nun zu den Pull-Down-Menüs und den darin vorhandenen Menüpunkten.

- Im Menü »Einlesen« befinden sich die Menüpunkte, die mit dem Einlesen des Diskettennamens zusammenhängen.

— Diskette Einlesen:

Das Programm liest den Diskettennamen der aktuellen Diskette und schiebt ihn vor dem aktuellen Eintrag in die Liste ein. Der Diskettenname wird auch in das Anzeigefenster übernommen, kann dort aber später editiert werden. Taste: <F1>.

— Autonumerierung:

Dieser nützliche Menüpunkt fragt zunächst nach einem Startwert. Danach schreibt das Programm bei jeder neu eingelesenen Diskette die fortlaufende Nummer, beginnend mit dem Startwert, in die Nummernspalte. Das ist insbesondere nützlich, wenn die Sammlung zum ersten Mal erfaßt wird, da man sich dann nicht mehr um die Nummern zu kümmern braucht. Die Autonumerierung wird durch ein zweites Anwählen des Menüpunkts wieder ausgeschaltet. Taste: <HELP>+<F5>.

— DF0:

Macht df0: zum aktuellen Laufwerk. Es wird später von diesem Laufwerk der Name gelesen. Taste: <F2>.

— DF1:

Macht df1: zum aktuellen Laufwerk. Taste: <F3>.

- Um die Übersichtlichkeit zu steigern, kann die Liste auch sortiert werden. Die benötigten Menüpunkte finden Sie im Menü »Sortieren«.

— Namen:

Es wird alphabetisch nach den Einträgen im Anzeigefenster

```
AmigaSort

ExtrasD

C-Disk                      0010
Extras (Kopie)              0004
Extras (Original)           0005
Juggler                     0009
M-Disk                      0013

df1:
```

Bild 1. Das Bedienungsfeld und die Ausgabefenster

```
Deluxe Paint II          0001 AmigaSort                    0002

Deluxe Paint II          0001    DPaint
AmigaSort                0002    AmigaSortDisk

Deluxe Paint II          0001    DPaint
                                   DPaint II und DPaint II PAL
                                   Bei der PAL-Version ist die
                                   vertikale Auflösung 256
                                   beziehungsweise 512 Punkte.
                                   Einige gute Bilder!
AmigaSort                0002    AmigaSortDisk
                                   AmigaSort dient zur
                                   Verwaltung von Disketten.
                                   Abgedruckt im Amiga-Magazin
                                   Ausgabe 10 1987.
                                   Sehr komfortabel!
```

Bild 2. Die drei verschiedenen Druckformate von AmigaSort

sortiert. Diese Funktion ist leider nicht sehr schnell. Bei 50 Einträgen muß man mit Zeiten von 20 bis 25 Sekunden rechnen. Taste: <F4>.

— Nummern:

Es wird nach Nummern sortiert, mit der kleinsten Nummer am Anfang. Der Zeitbedarf entspricht dem beim Sortieren nach Namen. Taste: <F5>.

• Die erfaßten Daten lassen sich jederzeit wieder verändern, indem im Menü »Editieren« die entsprechende Funktion angewählt wird.

— Name:

Hiermit kann der im Anzeigefenster angegebene Name geändert werden. Drückt man <HELP> und dann <RETURN>, so wird der Diskettenname übernommen. Taste: <F6>.

— Nummer:

Hier kann die im Nummernfenster angezeigte Nummer geändert werden. Damit sich selbsteingegebene Nummern mit den von der Autofunktion erzeugten vertragen, sollte man sie vierstellig mit führenden Nullen eingeben. Taste: <F7>.

— Kommentar:

Es wird ein Kommentar zum aktuellen Eintrag erzeugt. Man kann fünf Zeilen Text eingeben. Das Vorhandensein des Kommentars wird mit einem »K« im kleinen Fenster links oben vermerkt. Ist schon Text eingegeben worden, kann dieser Zeile für Zeile verändert werden. Will man eine Zeile übernehmen, drückt man <RETURN> und der alte Text bleibt erhalten. Taste: <F8>.

— Kommentar löschen:

Existiert zum aktuellen Eintrag ein Kommentar, so wird dieser gelöscht. Taste: <F9>.

— Eintrag löschen:

Der aktuelle Eintrag wird aus der Liste gelöscht. Taste: <F10>.

Das nächste Menü (»Liste«) enthält die Funktionen mit der Listen auf der Diskette verwaltet werden können.

— Liste laden:

Nach einer Sicherheitsabfrage fordert das Programm zur Eingabe eines Namens auf. Danach wird die Liste dieses Namens (mit dem Zusatz ».list«) geladen. Durch den Namen sind Sie natürlich in der Lage, mehrere verschiedene Listen mit AmigaSort zu verwalten. So könnten Sie zum Beispiel die Disketten nach verschiedenen Gebieten (Grafik, Musik und so weiter) aufteilen. Taste: <HELP> + <F10>.

— Liste speichern:

Die aktuelle Liste wird unter dem eingegebenen Namen (wieder Zusatz ».list«) gespeichert. Es kann hier natürlich wie auch beim Laden ein vollständiger Pfadname angegeben werden. Taste: <HELP> + <F2>.

— Liste erweitern:

Die aktuelle Liste wird an die Liste mit dem angegebenen Namen angehängt. Existiert eine Liste dieses Namens nicht, erzeugt das Programm eine neue Datei. Taste: <HELP> + <F3>.

— Liste löschen:

Nach einer Sicherheitsabfrage wird die aktuelle Liste aus dem Speicher gelöscht. Taste: <ESC>.

• Um aus dem Programm auszusteigen, gibt es das Menü »Beenden«.

Programmname:	Amiga Sort
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2

Fortsetzung der Beschreibung auf Seite 132

```
Programm : Amiga Sort
---------------------

 1 gF0 CLEAR ,60000&
 2 S0  GOSUB Init
 3 dm  ON MOUSE GOSUB Click
 4 9Q  MOUSE ON
 5 0G  Loop:
 6 tW2   SLEEP
 7 xs    x$=INKEY$
 8 Iv    IF x$<>"" THEN GOSUB Key ELSE GOTO
         Loop
 9 c9    GOTO Loop
10 j00 Key:
11 oZ2   MOUSE OFF
12 U0    MENU OFF
13 Bm    IF x$=CHR$(139) THEN GOSUB ChangeFlag

14 cA    IF x$="d" THEN GOSUB ShowName
15 tb    IF x$="k" THEN GOSUB Comment
16 EV    IF x$="-" THEN Position=1:GOSUB Show
17 3m    IF x$="+" THEN Position=Length:GOSUB
         Show
18 41    IF x$=CHR$(28) THEN Position=Position
         -1:GOSUB Show
19 1x    IF x$=CHR$(29) THEN Position=Position
         +1:GOSUB Show
20 xr    IF x$=CHR$(30) THEN Position=Position
         -5:GOSUB Show
21 un    IF x$=CHR$(31) THEN Position=Position
         +5:GOSUB Show
22 oW    IF x$=CHR$(129) THEN GOSUB Function1
23 Cn    IF x$=CHR$(130) THEN GOSUB Function2
24 Kx    IF x$=CHR$(131) THEN GOSUB Function3
25 S7    IF x$=CHR$(132) THEN GOSUB Function4
26 aH    IF x$=CHR$(133) THEN GOSUB Function5
27 iR    IF x$=CHR$(134) THEN GOSUB Function6
28 qb    IF x$=CHR$(135) THEN GOSUB Function7
29 yl    IF x$=CHR$(136) THEN GOSUB Function8
30 6v    IF x$=CHR$(137) THEN GOSUB Function9
31 ub    IF x$=CHR$(138) THEN GOSUB Function0
32 zV    IF x$=CHR$(27) THEN GOSUB Escape
33 ct    MOUSE ON
34 mC    MENU ON
35 mp    IF x$<>CHR$(139) THEN GOSUB ClearFl
         ag
36 wY    RETURN
37 cF0 Click:
38 dJ2   Dum=MOUSE(0)
39 4Z    IF INT(MOUSE(3)\30)=5 AND INT(MOUSE(4
         )\15)=2 THEN  Position=Position-1:GOT
         O Show
40 0U    IF INT(MOUSE(3)\30)=5 AND INT(MOUSE(4
         )\15)=3 THEN  Position=Position+1:GOT
         O Show
41 6G    IF INT((MOUSE(3)+10)\30)=4 AND INT(MO
         USE(4)\15)=2 THEN  Position=Position-
         5:GOTO Show
42 2B    IF INT((MOUSE(3)+10)\30)=4 AND INT(MO
         USE(4)\15)=3 THEN  Position=Position+
         5:GOTO Show
43 wu    IF INT((MOUSE(3)+20)\30)=3 AND INT(MO
         USE(4)\15)=2 THEN  Position=1:GOTO Sh
         ow
44 S8    IF INT((MOUSE(3)+20)\30)=3 AND INT(MO
         USE(4)\15)=3 THEN  Position=Length:GO
         TO Show
45 oE    IF INT((MOUSE(3)-10)\30)=6 AND INT(MO
         USE(4)\15)=2 THEN  GOTO Comment
46 6U    IF INT((MOUSE(3)-10)\30)=6 AND INT(MO
         USE(4)\15)=3 THEN  GOTO ShowName
47 7j    RETURN
48 se0 Init:
49 nV2   LINE (150,30)-(179,44),3,bf:AREA STEP
          (-4,-2):AREA STEP (-22,0):AREA STEP
         (11,-9):AREAFILL
50 CF    LINE (150,45)-(179,59),3,bf:AREA STEP
          (-4,-12):AREA STEP (-22,0):AREA STEP
         (11,9):AREAFILL
51 Ir    LINE (110,30)-(139,44),2,bf:AREA STEP
          (-4,-2):AREA STEP (-22,0):AREA STEP
         (11,-9):AREAFILL
52 hb    LINE (110,45)-(139,59),2,bf:AREA STEP
          (-4,-12):AREA STEP (-22,0):AREA STEP
         (11,9):AREAFILL
53 f0    LINE (70,30)-(99,44),2,bf:AREA STEP (
         -4,-2):AREA STEP (-22,0):AREA STEP (1
         1,-9):AREAFILL
54 DV    LINE (70,45)-(99,59),2,bf:AREA STEP (
         -4,-12):AREA STEP (-22,0):AREA STEP (
         11,9):AREAFILL
55 aG    LINE (72,32)-(97,32),1
56 UE    LINE (72,57)-(97,57),1
57 71    LINE (190,30)-(219,44),2,bf
58 fQ    LINE (190,45)-(219,59),2,bf
59 0p    LINE (198,34)-(198,40),1
60 Vt    LINE (211,34)-(198,37),1
61 95    LINE -(211,40),1
62 0Z    LINE (198,49)-(198,56),1
63 R3    CIRCLE (200,52),10,1,4.4,1.87
64 Lj    LINE (234,30)-(486,72),3,b
65 Qp    LINE (516,30)-(556,72),3,b
66 3p    LINE (234,12)-(486,26),3,b
67 8v    LINE (516,12)-(556,26),3,b
68 mF    LINE (149,61)-(184,73),3,b
69 8R    LINE (103,61)-(139,73),3,b
70 oJ    LOCATE 9,20
71 7Q    PRINT "df0:"
72 0f    FFlag=1
73 Mj    LIBRARY "dos.library"
74 Cp    LIBRARY "exec.library"
75 kE    DECLARE FUNCTION Lock& LIBRARY
76 8k    DECLARE FUNCTION Examine& LIBRARY
77 eV    DECLARE FUNCTION AllocMem& LIBRARY
78 XP    Drive$="df0:"
79 1a    Size&=252
80 Jy    Req&=2^1+2^16
81 4U    Info&=AllocMem&(Size&,Req&)
82 Ue    GOSUB Start
83 M8    Blank$=SPACE$(30)
84 jj    LBlank$=SPACE$(80)
85 1T    NBlank$=SPACE$(4)
86 PJ    DIM Disk$(400)
87 Na    DIM Nr$(400)
88 P7    DIM Label$(400)
89 VN    DIM CFlag$(400)
90 0K    DIM Comment$(400,5)
91 Pd    Position=1
92 ta    Length=0
93 GF0 Show:
94 0W2   IF Length=0 THEN RETURN
95 ks    IF Position<1 THEN Position=1
96 kz    IF Position>Length THEN Position=Len
         gth
97 Wi    IF Position=1 OR Position=2 THEN Item
```

Listing. »AmigaSort« zur Verwaltung von Disketten. Bitte mit dem Checksummer (Seite 66) eingeben.

```
        $(1)=Blank$:NItem$(1)=NBlank$ ELSE It
        em$(1)=Disk$(Position-2):NItem$(1)=Nr
        $(Position-2)
98 F4   IF Position=1 THEN Item$(2)=Blank$:NI
        tem$(2)=NBlank$ ELSE Item$(2)=Disk$(P
        osition-1):NItem$(2)=Nr$(Position-1)
99 Jh   Item$(3)=Disk$(Position):NItem$(3)=Nr
        $(Position)
100 QK  IF Position=Length THEN Item$(4)=Blan
        k$:NItem$(4)=NBlank$ ELSE Item$(4)=Di
        sk$(Position+1):NItem$(4)=Nr$(Positio
        n+1)
101 TB  IF Position=Length OR Position=Length
        -1 THEN Item$(5)=Blank$:NItem$(5)=NBl
        ank$ ELSE Item$(5)=Disk$(Position+2):
        NItem$(5)=Nr$(Position+2)
102 Sh  LOCATE 5,1
103 Qo  FOR x=1 TO 5
104 E54   PRINT SPC(30);Item$(x);SPC(5);NItem
          $(x)
105 c72 NEXT x
106 Ob  LOCATE 3,1
107 DN  PRINT SPC(30);Label$(Position);SPC(6)
        ;CFlag$(Position)
108 61  RETURN
109 pJ0 Comment:
110 A02  IF CFlag$(Position)<>"K" THEN RETUR
         N
111 bq   LOCATE 5,1
112 Zx   FOR x=1 TO 5
113 p14    PRINT SPC(30);Comment$(Position,x)
114 1G2  NEXT x
115 Dp   RETURN
116 Mu0 Start:
117 hw2  LOCATE 5,1
118 L9   PRINT SPC(30);"*********************
         *******"
119 z7   PRINT SPC(30);"********* AmigaSort **
         *******"
120 M7   PRINT SPC(30);"***** written July 198
         7 *****"
121 v4   PRINT SPC(30);"****** by Markus Säbel
          ******"
122 PD   PRINT SPC(30);"*********************
         *******"
123 sN   MENU 1,0,1,"Einlesen"
124 Yr   MENU 1,1,1,"Diskette einlesen"
125 k7   MENU 1,2,1,"  Autonummerierung"
126 OL   MENU 1,3,2,"  DF0:"
127 QT   MENU 1,4,1,"  DF1:"
128 2d   MENU 2,0,1,"Sortieren"
129 1w   MENU 2,1,1,"Namen"
130 sq   MENU 2,2,1,"Nummern"
131 SF   MENU 3,0,1,"Editieren"
132 zy   MENU 3,1,1,"Name"
133 dW   MENU 3,2,1,"Nummer"
134 Qb   MENU 3,3,1,"Kommentar"
135 nB   MENU 3,4,1,"Kommentar Löschen"
136 B9   MENU 3,5,1,"Eintrag Löschen"
137 z2   MENU 4,0,1,"Liste"
138 5e   MENU 4,1,1,"Liste laden"
139 ZU   MENU 4,2,1,"Liste speichern"
140 SY   MENU 4,3,1,"Liste erweitern"
141 Np   MENU 4,4,1,"Liste löschen"
142 LB   MENU 5,0,1,"Beenden"
143 QH   MENU 5,1,1,"Beenden"
144 77   MENU 6,0,1,"Drucken"
145 TZ   MENU 6,1,1,"Name/Nummer"
146 30   MENU 6,2,1,"Name/Diskette/Nummer"
147 8o   MENU 6,3,1,"Alle Daten"
148 oJ   MENU 7,0,1,"Suchen"
149 CV   MENU 7,1,1,"Namen"
150 yL   MENU 7,2,1,"Disketten"
151 RV   MENU 7,3,1,"Nummern"
152 Y9   ON MENU GOSUB Choose
153 h7   MENU ON
154 qS   RETURN
155 QaO DiskName:
156 oZ2  DName$=""
157 Py   count1=0
158 r8   Key$=Drive$+CHR$(0)
159 PX   ac%=-2
160 wN   Ex&=Lock&(SADD(Key$),ac%)
161 Tg   Dum&=Examine&(Ex&,Info&)
162 W20 NameGet:
163 jp2  Char$ = CHR$(PEEK(Info&+count1+8))
164 VO   IF ASC(Char$) <> 0 THEN DName$=DNam
         e$+Char$:count1=count1+1:GOTO NameGet

165 kS   DName$=DName$+STRING$(30-LEN(DName$),
         " ")
166 8X   CALL UnLock(Ex&)
167 3f   RETURN
168 J10 Choose:
169 1v2  MENU OFF
170 N8   MOUSE OFF
171 1p   ON MENU(0) GOSUB Einlesen,Sortieren,E
         ditieren,Liste,Beenden,Drucken,Suchen
         ,
172 OQ   MENU ON
173 s9   MOUSE ON
174 Am   RETURN
175 XTO Einlesen:
176 rB2  ON MENU(1) GOTO DEinlesen,AutoNr,Chan
         gedf0,Changedf1
177 fW4   DEinlesen:
178 fq6     IF Length=0 GOTO NewDisk
179 p8      FOR t=Length TO Position STEP -1
180 fU8       Disk$(t+1)=Disk$(t)
181 xB        Nr$(t+1)=Nr$(t)
182 XZ        Label$(t+1)=Label$(t)
183 d8        CFlag$(t+1)=CFlag$(t)
184 J7        FOR x=1 TO 5
185 vdA         Comment$(t+1,x)=Comment$(t,x)

186 vQ8       NEXT x
187 oF6     NEXT t
188 QW4   NewDisk:
189 OH6     GOSUB DiskName
190 Fd      Disk$(Position)=DName$
191 z1      Label$(Position)=DName$
192 CW      IF AFlag=0 THEN
193 Ze8       Nr$(Position)=NBlank$
194 jS6     ELSE
195 1G8       z$=STR$(AutoCount)
196 k6        y$=LEFT$(RIGHT$(z$,LEN(z$)-1),4
          )
197 sE        x$=LEFT$("0000",4-LEN(y$))+y$
198 2t        Nr$(Position)=LEFT$(x$+NBlank$,
          4)
199 H2        AutoCount=AutoCount+1
200 A36     END IF
201 JX      CFlag$(Position)=" "
202 1P      FOR x=1 TO 5
203 H08       Comment$(Position,x)=Blank$
204 D16     NEXT x
205 A8      Length=Length+1
206 Ft      GOTO Show
207 fy4   AutoNr:
208 Sm6     IF AFlag=0 THEN
209 3d8       AFlag=1
210 sa        MENU 1,2,2
211 bz        LOCATE 1,21
212 rV        LINE INPUT "Startwert:";N$
213 zA        LOCATE 1,1
214 xP        PRINT LBlank$
215 w5        AutoCount=VAL(N$)
216 5o7     ELSE
217 8h9       AFlag=0
218 ug        MENU 1,2,1
219 TM7     END IF
220 uW      RETURN
221 Nk4   Changedf0:
222 rj6     Drive$="df0:"
223 9s      MENU 1,3,2
224 8w      MENU 1,4,1
225 Jo      LOCATE 9,20
226 f0      PRINT Drive$
227 1d      RETURN
228 Zx4   Changedf1:
229 Ot6     Drive$="df1:"
230 Ax      MENU 1,3,1
231 L5      MENU 1,4,2
232 Qv      LOCATE 9,20
233 m7      PRINT Drive$
234 8k      RETURN
235 q22  Sortieren:
236 Gx4   ON MENU(1) GOTO NameS,NummerS
237 126     NameS:
238 9J8       y=0
239 MA        FOR t=1 TO Length-1
240 9RA         IF Disk$(t)>Disk$(t+1) THEN
              GOSUB Change
241 g78       NEXT t
242 oT        IF y=1 GOTO NameS
243 qU        GOTO Show
244 Iu        RETURN
245 Cu6     NummerS:
246 HR8       y=0
247 UI        FOR t=1 TO Length-1
248 oOA         IF Nr$(t)>Nr$(t+1) THEN GOSU
                B Change
249 oF8       NEXT t
250 5d        IF y=1 GOTO NummerS
251 yc        GOTO Show
252 Q2        RETURN
253 ZD2 Editieren:
254 wV4   ON MENU(1) GOTO EName,ENummer,EKomm
       entar,KLoeschen,ELoeschen
255 S36     EName:
256 If8       LOCATE 1,20
257 S6        LINE INPUT "Neuer Name:";N$
258 it        LOCATE 1,1
259 g8        PRINT LBlank$
260 9y        IF N$="" THEN RETURN
261 zn        IF ASC(N$)=139 THEN Disk$(Posit
              ion)=Label$(Position):GOTO Show

262 gp        Disk$(Position)=LEFT$(N$+Blank$
              ,30)
263 Ao        GOTO Show
264 hX6     ENummer:
265 r68       LOCATE 1,54
266 iy        LINE INPUT "Neue Nummer:";N$
267 r2        LOCATE 1,1
268 pH        PRINT LBlank$
269 I7        IF N$="" THEN RETURN
270 Dc        Nr$(Position)=LEFT$(N$+Blank$,4
              )
271 Iw        GOTO Show
272 rb6     EKommentar:
273 DS8       LOCATE 5,1
274 nx        IF NOT CFlag$(Position)="K" THE
              N
275 ytA         CFlag$(Position)="K"
276 Db          FOR x=1 TO 5
277 TaC           Comment$(Position,x)=Blank$

278 PuA         NEXT x
279 RK8       END IF
280 Hf        FOR x=1 TO 5
281 XjA         PRINT SPC(30);Comment$(Positi
                on,x)
282 Ty8       NEXT x
283 wa        FOR t=1 TO 5
284 sDA         LOCATE 3+t,1
285 uL          PRINT SPC(28);" "
286 yK          LOCATE 4+t,1
287 1U          PRINT SPC(28);CHR$(187)
288 qE          LOCATE 1,21
289 Ns          LINE INPUT "Kommentar:";N$
290 EP          LOCATE 1,1
291 Ce          PRINT LBlank$
292 gJ          IF N$="" GOTO UpdateLine:
293 Ys          Comment$(Position,t)=LEFT$(N$
                +Blank$,30)
294 x38       UpdateLine:
295 7TA         LOCATE 4+t,1
296 We          PRINT SPC(30);Comment$(Positi
                on,t)
297 a18       NEXT t
298 sB        LOCATE 9,1
299 8Z        PRINT SPC(28);" "
300 1P        GOTO Show
301 yL6     KLoeschen:
302 wA8       CFlag$(Position)=" "
303 e2        FOR x=1 TO 5
304 BIA         Comment$(Position,x)=""
305 qL8       NEXT x
306 rV        GOTO Show
307 s96     ELoeschen:
308 qy8       IF Length=0 THEN RETURN
309 Vi        IF Length=1 GOTO ClearAll
310 1w        FOR t=Position TO Length-1
311 beA         Disk$(t)=Disk$(t+1)
```

```
312 S4             Nr$(t)=Nr$(t+1)
313 1Q             Label$(t)=Label$(t+1)
314 Gb             CFlag$(t)=CFlag$(t+1)
315 qE             FOR x=1 TO 5
316 cuC              Comment$(t,x)=Comment$(t+1,
                     x)
317 2XA            NEXT x
318 vM8          NEXT t
319 AA           Length=Length-1
320 5j           GOTO Show
321 MQ6        ClearAll:
322 208          Disk$(1)=Blank$
323 0k           Nr$(1)=NBlank$
324 1q           Label$(1)=Blank$
325 ZF           CFlag$(1)=" "
326 1P           FOR x=1 TO 5
327 8EA            Comment$(1,x)=Blank$
328 Di8          NEXT x
329 b7           GOSUB Show
330 JQ           Length=0
331 hJ           RETURN
332 xV2    Liste:
333 vR4      ON MENU(1) GOTO LLoad,LSave,LAppend
             ,LLoeschen
334 hT2    LLoad:
335 1z4      GOSUB Sicherheitsabfrage
336 DO       IF Shure=0 THEN RETURN
337 er       Counter=0
338 mE       LOCATE 1,25
339 ok       LINE INPUT "Liste:";N$
340 2D       LOCATE 1,1
341 OS       PRINT LBlank$
342 TI       IF N$="" THEN RETURN
343 CP       OPEN LEFT$(N$,25)+".list" FOR INPUT
              AS 1
344 qx6        WHILE EOF(1)=0
345 ZF8          Counter=Counter+1
346 Nu           LINE INPUT#1,Disk$(Counter)
347 XX           LINE INPUT#1,Label$(Counter)
348 9W           LINE INPUT#1,Nr$(Counter)
349 2e           LINE INPUT#1,CFlag$(Counter)
350 iM           IF CFlag$(Counter)="K" THEN
351 QoA            FOR x=1 TO 5
352 gsC              LINE INPUT#1,Comment$(Count
                     er,x)
353 c7A            NEXT x
354 eX8          END IF
355 Ay6        WEND
356 yl4      CLOSE #1
357 dT       Length=Counter
358 iw       Position=1
359 iM       GOTO Show
360 LH2    LSave:
361 9b4      LOCATE 1,25
362 B7       LINE INPUT "Liste:";N$
363 Pa       LOCATE 1,1
364 Np       PRINT LBlank$
365 qf       IF N$="" THEN RETURN
366 zX       OPEN LEFT$(N$,25)+".list" FOR OUTPU
             T AS 1
367 DX6        FOR t=1 TO Length
368 R28          PRINT #1,Disk$(t)
369 w4           PRINT #1,Label$(t)
370 zS           PRINT #1,Nr$(t)
371 3L           PRINT #1,CFlag$(t)
372 zR           IF CFlag$(t)="K" THEN
373 mAA            FOR x=1 TO 5
374 LbC              PRINT#1,Comment$(t,x)
375 yTA            NEXT x
376 0t8          END IF
377 sJ6        NEXT t
378 K74      CLOSE #1
379 T5       RETURN
380 v42    LAppend:
381 Tv4      LOCATE 1,25
382 VR       LINE INPUT "Liste:";N$
383 Ju       LOCATE 1,1
384 h9       PRINT LBlank$
385 Az       IF N$="" THEN RETURN
386 HT       OPEN LEFT$(N$,25)+".list" FOR APPEN
             D AS 1
387 Xr6        FOR t=1 TO Length
388 1M8          PRINT #1,Disk$(t)
389 GO           PRINT #1,Label$(t)
390 Jm           PRINT #1,Nr$(t)
391 Nf           PRINT #1,CFlag$(t)
392 J1           IF CFlag$(t)="K" THEN
393 6UA            FOR x=1 TO 5
394 fvC              PRINT#1,Comment$(t,x)
395 InA            NEXT x
396 KD8          END IF
397 Cd6        NEXT t
398 eR4      CLOSE #1
399 nP       RETURN
400 bz2    LLoeschen:
401 p34      GOSUB Sicherheitsabfrage
402 HS       IF Shure=0 THEN RETURN
403 n7       FOR t=1 TO Length
404 VY6        Disk$(t)=""
405 up         Label$(t)=""
406 1X         Nr$(t)=""
407 YO         IF CFlag$(t)="K" THEN
408 LJ8          FOR x=1 TO 5
409 qEA            Comment$(t,x)=""
410 hm8          NEXT
411 ZS6        END IF
412 Sv         CFlag$=""
413 St4      NEXT t
414 cq       Position=1
415 As       Length=1
416 nj       GOTO ELoeschen
417 Ab2    Beenden:
418 6K4      GOSUB Sicherheitsabfrage
419 Yj       IF Shure=0 THEN RETURN
420 Qe       LIBRARY CLOSE
421 XS       END
422 142    Drucken:
423 YA4      ON MENU(1) GOTO DiskNr,DiskLabelNr,
             PrintAll
424 IN2    DiskNr:
425 Ld4      FOR t=1 TO INT(Length/2+.5)
426 JE6        LPRINT Disk$(t);" ";Nr$(t);" ";
427 sv         LPRINT Disk$(INT(t+Length/2+.5));
               " ";Nr$(INT(t+Length/2+.5))
428 h84      NEXT t
429 Ht       RETURN
430 WD2    DiskLabelNr:
431 FZ4      FOR t=1 TO Length
432 486        LPRINT Disk$(t);" ";Nr$(t);"    "
               ;Label$(t)
433 mD4      NEXT t
434 My       RETURN
435 Pv2    PrintAll:
436 Ke4      FOR t=1 TO Length
437 9D6        LPRINT Disk$(t);" ";Nr$(t);"    "
               ;Label$(t)
438 3V         IF CFlag$(t)="K" THEN
439 qE8          FOR x=1 TO 5
440 BLA            LPRINT TAB(42);Comment$(t,x)
441 2X8          NEXT x
442 4x6        END IF
443 wN4      NEXT t
444 W8       RETURN
445 WB2    Suchen:
446 3Z4      ON MENU(1) GOTO SName,SLabel,SNumme
             r
447 Op2    SName:
448 aA4      LOCATE 1,19
449 oG       LINE INPUT "Suchbegriff:";N$
450 oz       LOCATE 1,1
451 mE       PRINT LBlank$
452 F4       IF N$="" THEN RETURN
453 94       t=Position+1
454 f12    DLoop:
455 VW4      IF INSTR(Disk$(t),N$)<>0 THEN Pos
             ition=t:GOTO Show
456 Qg       t=t+1
457 4Y       IF t<=Length THEN DLoop
458 X3       BEEP
459 1N       RETURN
460 6w2    SLabel:
461 nN4      LOCATE 1,19
462 1T       LINE INPUT "Suchbegriff:";N$
463 1C       LOCATE 1,1
464 zR       PRINT LBlank$
465 SH       IF N$="" THEN RETURN
466 MH       t=Position+1
467 8M2    LLoop:
468 Cv4      IF INSTR(Label$(t),N$)<>0 THEN Po
             sition=t:GOTO Show
469 dt       t=t+1
470 vX       IF t<=Length THEN LLoop
471 kG       BEEP
472 ya       RETURN
473 Wa2    SNummer:
474 Oa4      LOCATE 1,19
475 Eg       LINE INPUT "Suchbegriff:";N$
476 EP       LOCATE 1,1
477 Ce       PRINT LBlank$
478 fU       IF N$="" THEN RETURN
479 ZU       t=Position+1
480 Pf2    NLoop:
481 qJ4      IF Nr$(t)=N$ THEN Position=t:GOTO S
             how
482 q6       t=t+1
483 Iw       IF t<=Length THEN NLoop
484 xT       BEEP
485 Bn       RETURN
486 uQ0    ShowName:
487 o52      GOSUB DiskName
488 Y1       LOCATE 3,1
489 aZ       PRINT SPC(30);DName$
490 Gs       RETURN
491 NMO    Change:
492 SF2      SWAP Disk$(t),Disk$(t+1)
493 rD       SWAP Nr$(t),Nr$(t+1)
494 pH       SWAP CFlag$(t),CFlag$(t+1)
495 M8       SWAP Label$(t),Label$(t+1)
496 ss       IF CFlag$(t)="K" OR CFlag$(t+1)="K" T
             HEN
497 mA4        FOR x=1 TO 5
498 UW6          SWAP Comment$(t,x),Comment$(t+1,x
             )
499 yT4        NEXT x
500 Ot2      END IF
501 Sd       y=1
502 S4       RETURN
503 OIO    ChangeFlag:
504 j42      IF FFlag=1 THEN
505 2i4        FFlag=2
506 Sr         LINE (105,62)-(137,72),1,bf
507 mV2      ELSE
508 2h4        FFlag=1
509 Qo         LINE (105,62)-(137,72),0,bf
510 A32      END IF
511 bD       RETURN
512 230    ClearFlag:
513 7m2      FFlag=1
514 Vt       LINE (105,62)-(137,72),0,bf
515 fH       RETURN
516 KXO    Function1:
517 5g2      ON FFlag GOTO DEinlesen,LLoad
518 RfO    Function2:
519 uw2      ON FFlag GOTO Changedf0,LSave
520 YnO    Function3:
521 VT2      ON FFlag GOTO Changedf1,LAppend
522 fvO    Function4:
523 8C2      ON FFlag GOTO NameS,DiskNr
524 m30    Function5:
525 1E2      ON FFlag GOTO NummerS,AutoNr
526 tBO    Function6:
527 EP2      ON FFlag GOTO EName,DiskLabelNr
528 OJO    Function7:
529 GJ2      ON FFlag GOTO ENummer,PrintAll
530 7RO    Function8:
531 DM2      ON FFlag GOTO EKommentar,SName
532 EZO    Function9:
533 TC2      ON FFlag GOTO KLoeschen,SLabel
534 XjO    Function0:
535 Hp2      ON FFlag GOTO ELoeschen,SNummer
536 e50    Escape:
537 Wd2      ON FFlag GOTO LLoeschen,Beenden
538 OAO    Sicherheitsabfrage:
539 zJ2      LOCATE 1,31
540 1c       PRINT "Sicher?(J/N)"
541 UKO    KeyLoop:
542 aV2      x$=INKEY$
543 mK       IF x$="" THEN KeyLoop
544 tN       IF x$="j" OR x$="J" THEN Shure=1 ELSE
              Shure=0
545 5P       LOCATE 1,31
546 7o       PRINT Blank$
547 Bn       RETURN
(C) 1987 M&T
```

»AmigaSort« dient zur Verwaltung von Disketten

Nie wieder Tippfehler!

Endlich ist er da, der Checksummer für den Amiga! Die Zeiten der endlosen Fehlersuche sind damit vorbei. »Checkie42« funktioniert für alle Programmiersprachen von Basic über C bis zum Assembler, wenn sie im ASCII-Format vorliegen. Er ist also universell einsetzbar und genügt allen Ansprüchen.

Damit wirklich jeder Leser unseren neuen Checksummer benutzen kann, wurde er in Amiga-Basic geschrieben. Trotz der sehr hohen Zuverlässigkeit und der strukturierten Programmierung konnte er recht kurz gehalten werden. Dabei bietet »Checkie42« aber allen nur erdenklichen Komfort und Sicherheit.

Nach dem Abtippen (siehe Listing), bei dem Sie diesmal noch gut aufpassen müssen, haben Sie endlich die erwünschte Hilfe beim Abtippen unserer Programme. Ab dieser Ausgabe werden alle Listings mit den Checksummen abgedruckt. Bevor wir zur Bedienung des Programms übergehen, soll zuvor noch erläutert werden, aus welchen Teilen sich eine Zeile in Zukunft zusammensetzt. Ein Beispiel für eine solche Zeile sieht so aus:

```
1 TTO print "Hallo!"
```

Die erste Zahl (»1«) stellt die Zeilennummer dar. Die Werte hierfür reichen von 1 bis maximal 9999. Nach einer Leerstelle folgt dann die eigentliche Prüfziffer, die sich noch mal unterteilt. Die ersten zwei Zeichen sind die Prüfziffern für den Text und die Zeilennummer. Erlaubt sind hier die Zeichen von »0« bis »9«, die großen (»A« bis »Z«) und die kleinen Buchstaben (»a« bis »z«). Durch die große Anzahl von verschiedenen Zeichen wurde erreicht, daß eine zweistellige Prüfsumme vollkommen ausreicht. Um dieselbe Fehlersicherheit mit Ziffern allein zu erreichen, würde man mindestens fünf bis sechs Stellen benötigen.

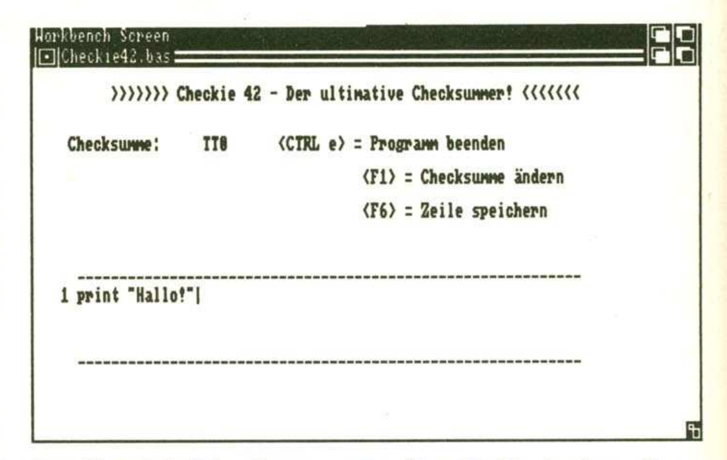
Das letzte Zeichen der Checksumme ist interessant, wenn die abgedruckten Listings strukturiert aufgebaut sind. Die dritte Stelle tritt nur dann auf, wenn sich der Beginn der Zeile verschoben hat. Deshalb erscheint sie auch nicht immer. Nach einer weiteren Leerstelle folgt dann endlich der eigentliche Text.

Doch nun zur Bedienung von »Checkie42«. Nachdem das Programm gestartet wurde, sehen Sie ein Fenster auf dem Bildschirm. In diesem Fenster erscheint zunächst eine Abfrage nach der gewünschten Datei. Hier können Sie einen neuen Dateinamen oder den Namen einer bereits bestehenden Datei eingeben. Zu letzterem Punkt kommen wir später.

Nach der Eingabe, die mit <RETURN> abgeschlossen wird, kommt nun das eigentliche Arbeitsfenster von »Checkie42« (siehe Bild). Links oben blinkt der Text »Checksumme:« und fordert Sie zur Eingabe der im Heft abgedruckten Prüfzeichen auf. Wenn Sie Wert darauf legen, daß das Listing auch bei Ihnen mit den richtigen Einrückungen generiert wird, müssen Sie auch die eventuell vorhandene dritte Stelle mit eingeben. Ansonsten geben Sie nur die ersten zwei Zeichen ein und drücken dann <RETURN>. Ist die dritte Stelle nicht vorhanden, beenden Sie die Eingabe auch mit <RETURN>.

Der Cursor steht nun zwischen den zwei Trennstrichen in der Mitte des Bildschirms. Davor hat das Programm automatisch die Zeilennummer geschrieben. Außerdem wurde der Cursor schon richtig positioniert, wenn die dritte Prüfziffer mit eingegeben wurde. Zum Beenden drücken Sie einfach die RETURN-Taste. Bei fehlerfreier Zeile wird der Raum zum Editieren gelöscht, die nächste Zeilennummer wird angezeigt und das Programm erwartet die Eingabe der nächsten Checksumme.

Leider wird es auch vorkommen, daß der Text einen Fehler enthält. Wenn Sie diesen entdeckt haben, können Sie den Cursor, der ganz am Anfang des Textes steht, an die entsprechende Stelle bewegen. Dabei verwenden Sie die Cursor-Steuertasten, um die Position zu erreichen. Mit der Backspace-Taste löschen Sie dann das Zeichen rechts vom Cursor. Auch das Einfügen von Zeichen ist ohne weiteres möglich. Dazu drücken Sie <TAB>, worauf an der Cursor-Position eine Leerstelle eingefügt wird. Dort können Sie nun das fehlende Zeichen eintragen. Für mehrere Zeichen muß entsprechend oft <TAB> gedrückt werden.



Das übersichtliche Fenster von »Checkie42«, in dem alle Eingaben gemacht und editiert werden können

Wenn Sie die Zeile trotz falscher Checksumme übernehmen wollen, drücken Sie die Taste <F6>. Dies ist zum Beispiel sinnvoll, wenn Sie Kommentare nicht mit eingeben wollen. Natürlich kann es auch vorkommen, daß Sie zwar die Zeile richtig, aber die Prüfsumme falsch eingegeben haben. In diesem Fall können Sie jederzeit die Taste <F1> drücken, worauf Sie die Checksumme ändern können.

Checkie42 speichert die einzelnen Zeilen nach Drücken von <RETURN> oder <F6> sofort in die Datei. Am Schluß des Listings müssen Sie dem Programm aber noch mitteilen, daß das Ende erreicht wurde. Das erreichen Sie durch Drücken von <CTRL E>. Wenn Sie Zeilen trotz falscher Prüfsumme übernommen haben, erfolgt eine Anzeige, wie viele Zeilen es waren. Längere Listings können Sie auch in Etappen abtippen. Wenn Sie keine Lust oder Zeit mehr haben, beenden Sie das Programm mit <CTRL E>. Bei dem nächsten Start geben Sie dann den gleichen Dateinamen ein und Checkie42 sucht automatisch diese Datei. Wenn sie gefunden wurde, lädt das Programm alle Zeilen und ermittelt so die nächste Zeilennummer. Ab dieser Position können Sie dann den Quelltext weiter eingeben.

Noch eine interessante Einzelheit von Checkie42 ist vielleicht für Sie von Interesse: Leerstellen vor und hinter dem eigentlichen Text werden nicht mitgeprüft. Wollten Sie also die Einrückung einer Zeile verändern, so stellt dies kein Problem dar.

Für diejenigen, denen die Basic-Version zu langsam sein sollte, noch ein kleiner Tip. Die Berechnung der eigentlichen Prüfsumme geschieht ab dem Label »CalcSumme:«. Dieser Teil ist sehr leicht in andere Programmiersprachen wie zum Beispiel C umsetzbar.

Nach dem Abtippen sollten Sie Checkie42 testen, indem Sie eines der in dieser Ausgabe abgedruckten Listings abtippen. Probieren Sie jede Funktion des Programmes durch, um etwaige Fehler durch das Abtippen zu entdecken. Wenn Ihr Checksummer dann richtig läuft, brauchen Sie in Zukunft keine Angst mehr vor Tippfehlern zu haben. Sie sparen damit viel Zeit für die vielleicht vergebliche Suche von tückischen Fehlern.

(Dieter Behlich/pa/rb)

Programmname:	Checkie42
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2

```
Programm : Checkie42
--------------------

 1 REM ***********************************
 2 REM *** Checksummer: Checkie 42     ***
 3 REM ***         Version 1.0         ***
 4 REM ***********************************
 5 Start:
 6   GOSUB Init
 7   GOSUB OpenDatei
 8   GOSUB Bild
 9 NeueZeile:
10   GOSUB Loeschen
11   GOSUB EingabeSumme
12   IF FEnde=wahr THEN Ende
13 Wiederholung:
14   GOSUB EingabeZeile
15   IF FEnde=wahr THEN Ende
16   GOSUB CalcSumme
17   IF FSumme=falsch THEN Wiederholung
18   GOSUB Uebernahme
19   GOTO NeueZeile
20 Ende:
21   GOSUB fertig
22   END
23
24 Init:
25   wahr=-1
26   falsch=0
27   LZeile=240  : REM Anzahl Zeichen/Zeile
28   LBZeile=60  : REM Anzahl Zeichen/Bildschrimzeile
29   AnzBZeilen=LZeile/LBZeile
30   zx=6 : zy=14   : REM Position Zeile
31   sx=20 : sy=5   : REM Position Checksumme
32   AnzCsZ=3       : REM Anzahl Ziffern/Checksumme
33   DIM z(LZeile)
34   DIM cs(AnzCsZ)
35   cs(AnzCsZ)=0   : REM Zeilenstart
36   a=0 : b=0 : c=0  : REM Hilfsvariablen
37   i=0 : j=0 : k=0  : REM Zählvariablen
38   READ Faktor(i)
39   WHILE Faktor(i)<>0
40     i=i+1
41     READ Faktor(i)  : REM Faktorenreihe
42   WEND
43   AnzFak=i
44   DATA 2,3,4,5,6,0
45   RETURN
46
47 OpenDatei:
48   ON ERROR GOTO Dateifehler
49   CLS
50   LOCATE 3,5
51   dn$="":INPUT "Dateiname: ";dn$
52   Zeile=1
53   OPEN dn$ FOR INPUT AS #1
54   IF FFehler=53 THEN NeueDatei
55   WHILE NOT EOF(1)
56     LINE INPUT#1,e$
57 REM PRINT e$
58     Zeile=Zeile+1
59   WEND
60   CLOSE 1
61 NeueDatei:
62   OPEN dn$ FOR APPEND AS #1
63   RETURN
64 Dateifehler:
65   FFehler=ERR
66   IF FFehler=53 THEN RESUME NEXT
67   ON ERROR GOTO 0
68
69 Bild:
70   CLS
71   LOCATE 2,10
72   PRINT ">>>>>>> Checkie 42 - Der ultimative Checksummer!
     <<<<<<<"
73   LOCATE 13,zx
74   FOR i=1 TO 2
75     FOR j=1 TO LBZeile
76       PRINT "-";
77     NEXT j
78     LOCATE 13+AnzBZeilen+1,zx
79   NEXT i
80   LOCATE 5,30
81   PRINT "<CTRL e> = Programm beenden"
82   LOCATE 7,40
83   PRINT "<F1> = Checksumme ändern"
84   LOCATE 9,40
85   PRINT "<F6> = Zeile speichern"
86   RETURN
87
```

Listing. Der ASCII-Checksummer »Checkie42« verhindert Fehler beim Abtippen von Listings. Bitte ohne Zeilennummern eingeben.

```
88 Loeschen:
89   LOCATE zy,1
90   FOR i=1 TO AnzBZeilen
91     PRINT TAB(zx);SPACE$(LBZeile)
92   NEXT i
93   FOR i=0 TO LZeile
94     z(i)=32
95   NEXT i
96   LOCATE zy,1
97   PRINT USING "####";Zeile
98   aPos=0
99   RETURN
100
101 EingabeSumme:
102   Farbe=0 : i=1
103   LOCATE sy,sx+1
104   PRINT "..."
105 blinken:
106   Farbe=Farbe XOR 1
107   COLOR Farbe,0
108   LOCATE sy,sx-15
109   PRINT "Checksumme:"
110   e$=INKEY$
111   IF e$="" THEN blinken
112   e=ASC(e$)-48
113   IF e=-43 THEN
114     FEnde=wahr   : REM <CTRL e>
115     GOTO ESEnde
116   ELSE
117     IF e<>-35 THEN keinEnde   : REM <CR>
118     IF i<AnzCsZ THEN blinken
119     GOTO ESEnde
120   keinEnde:
121     IF e<0 THEN blinken
122     IF e>9 THEN e=e-7
123     IF e<0 THEN blinken
124     IF e>35 THEN e=e-6
125     IF e<0 THEN blinken
126     IF e>61 THEN blinken
127     LOCATE sy,sx+i
128     COLOR 1,0
129     PRINT e$;
130     cs(i)=e
131     i=i+1
132     IF i<=AnzCsZ THEN blinken
133   END IF
134 ESEnde:
135   COLOR 1,0
136   LOCATE sy,sx-15
137   PRINT "Checksumme:"
138   RETURN
139
140 EingabeZeile:
141   x=cs(AnzCsZ)
142 weiter:
143   LOCATE zy+INT(x/LBZeile),zx+(x MOD LBZeile)
144   IF x>aPos THEN aPos=x
145   e$=INPUT$(1)
146   e=ASC(e$)
147   IF (e AND 127)<32 THEN Controlcode
148   PRINT e$
149   z(x)=e : e=30
150 Controlcode:
151   IF e=13 THEN
152     RETURN
153   ELSEIF e=30TTHEN
154     a=1
155   ELSEIF e=29 THEN
156     a=LBZeile
157   ELSEIF e=31 THEN
158     a=-1
159   ELSEIF e=28 THEN
160     a=-LBZeile
161   ELSE
162     GOTO noCr
163   END IF
164   x=x+a
165   IF x>=0 AND x<LZeile THEN weiter
166   x=x-a
167   GOTO weiter
168 noCrs:
169   IF e=8 THEN
170     FOR i=x TO aPos
171       z(i)=z(i+1)
172       PRINT CHR$(z(i));
173      IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
174     NEXT i
175     z(aPos)=32 : PRINT " "
176     aPos=aPos-1
177   ELSEIF e=9 THEN
178     IF aPos>x THEN
179       FOR i=aPos TO x STEP -1
180         z(i+1)=z(i)
181       NEXT i
182       z(x)=32
183       aPos=aPos+1
184       IF aPos=LZeile THEN aPos=aPos-1:z(LZeile)=32
185       FOR i=x TO aPos
186         PRINT CHR$(z(i));
187         IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
188       NEXT i
189      END IF
190   ELSEIF e=129 THEN
191     GOSUB EingabeSumme
192     x=cs(AnzCsZ)
193   ELSEIF e=134  THEN
194     RETURN
195   ELSEIF e=5 THEN
196     FEnde=wahr
197     RETURN
198   END IF
199   GOTO weiter
200
201 CalcSumme:
202   a=0 : b=0 : c=0
203   IF e=134 THEN
204     FSumme=wahr
205     FF6=FF6+1
206   ELSE
207     WHILE z(aPos)=32 AND aPos>0
208       aPos=aPos-1
209     WEND
210     IF aPos>0 THEN
211       WHILE z(c)=32
212         c=c+1
213       WEND
214     END IF
215     FOR i=c TO aPos
216       j=(i-c) MOD AnzFak
217       k=(i+1-c) MOD AnzFak
218       a=a+((z(i) AND 127)-32)*Faktor(j)
219       b=b+((z(i) AND 127)-32)*Faktor(k)
220     NEXT i
221     FSumme=(cs(1)=(a+Zeile) MOD 62) AND (cs(2)=(b+Zeile) MOD 62)
222   END IF
223   RETURN
224
225 Uebernahme:
226   FOR i=0 TO aPos
227     PRINT#1,CHR$(z(i));
228   NEXT i
229     PRINT#1,""
230   Zeile=Zeile+1
231   RETURN
232
233 fertig:
234   CLOSE 1
235   CLS
236   LOCATE 12,35
237   PRINT "F E R T I G !!!"
238   LOCATE 20,1
239   IF FF6<>0 THEN
240     PRINT "ACHTUNG!!!  ";
241     PRINT FF6;" Zeile(n) wurde(n) ungeprÜft gespeichert."
242   END IF
243   RETURN
(C) 1987 M&T
```

Listing. Der ASCII-Checksummer »Checkie 42« verhindert Fehler beim Abtippen von Listings (Schluß)

Jahres-Inhaltsverzeichnis

Auch in unserem AMIGA-Magazin wollen wir die Tradition weiterführen und zur Januar-Ausgabe das Jahres-Inhaltsverzeichnis veröffentlichen. Es ist nach Rubriken geordnet und mit einem Stichwort zum schnelleren Suchen versehen. Alle Artikel, die mit einem Stern gekennzeichnet wurden, sind auf den Programmservice-Disketten enthalten. Zusätzlich finden Sie am Schluß noch eine Übersicht aller Kurse, die bisher im AMIGA-Magazin veröffentlicht wurden. (do)

DAS AMIGA GLOSSAR TEIL 5

Die einzelnen Teile des Glossars zum Heraustrennen bilden ein umfassendes Nachschlagewerk der gängigsten Begriffe rund um den Amiga. Sammeln Sie mit.

Was verbirgt sich hinter den Begriffen Multitasking oder Multiprocessing? Das Amiga-Glossar hat eine Antwort auf diese und andere Fragen rund um den Amiga. Die Begriffe sind nach englischen Stichworten geordnet. Das erleichtert Ihnen die Arbeit mit der englischen Systemdokumentation.

(David Twigg-Flesner/pa)

linked list (verkettete Liste) Eine Liste enthält Einträge in einer bestimmten Reihenfolge (Beispiel: Adressen alphabetisch sortiert). Beim Einfügen eines neuen Elements (hier Adresse) entsprechend der gewählten Sortierordnung müßten alle folgenden Elemente um einen Platz verschoben werden. Eine verkettete Liste enthält numerierte Einträge in beliebiger Folge. Um dennoch eine sortierte Reihenfolge zu gewährleisten, enthält jeder Eintrag einen Verweis (Zeiger) auf die Nummer des in der Sortierfolge nächsten Eintrags. Der Zeiger des nach der Sortierfolge letzten Eintrages (er kann sich irgendwo innerhalb der Liste befinden) ist meist Null. Neue Einträge können nun einfach an das Ende der Liste angefügt werden. Statt einer Verschiebung werden lediglich die Zeiger korrigiert.

Das Prinzip der verketteten Liste spielt auf allen Ebenen des Betriebssystems des Amiga eine zentrale Rolle. Eine solche verkettete Liste besteht aus einem Listheader, dessen Adresse als Bezugspunkt für die ganze Liste gilt und einer unterschiedlichen, wechselnden Anzahl von »nodes« (Einträgen). Die Adresse des derzeit ersten und letzten node wird im Header abgelegt. Die einzelnen nodes enthalten Zeiger auf die Adressen der unmittelbaren Nachbareinträge (nach vorn und hinten), den Datentyp, die Priorität und den Inhalt des Eintrags.

Mit den entsprechenden Verwaltungsroutinen von -> exec lassen sich diese Listen nach dem LIFO-(last in, first out) oder FIFO-(first in, first out) Prinzip gestalten. Auch nach Priorität (zum Beispiel bei Tasks) verkettete Listen sind möglich, werden dann aber unter Umständen -> queues genannt. Mit den verschiedenen Listentypen führen sowohl Exec als auch Intuition Buch über das Geschehen im Computer.

load file (lauffähige Programmdatei) -> link editor

lock -> file lock, filing system

lockup (Aussperrung) Eine besondere Form des Absturzes. Der Computer weigert sich, weitere Eingaben anzunehmen. Wahrscheinliche Ursachen beim Amiga können sein: a) ein -> deadlock oder deadly embrace. b) fehlerhafte Programmierung (das Programm steckt in einer Endlosschleife). c) disk swap lockup: nach dem Kopieren einer Diskette und dem Versuch, mit der Kopie weiterzuarbeiten, verlangt der Computer abwechselnd Original und Kopie. Die Ursache des Fehlverhaltens ist meistens nicht zu ergründen, vermutlich aber ein Konflikt in der Suche nach -> logical devices oder fehlerhafte Übereinstimmung zwischen den Daten im Diskpuffer und den Daten auf der Diskette (»dirty track«). d) Speicherplatz erschöpft — wird meist durch einen Gruß vom -> Guru begleitet. Im ungünstigen Fall wird der Videospeicher beeinträchtigt und der Totalabsturz erfolgt ohne weitere Warnung. e) Adresse des -> message port für das aktuelle Window durch falsch abgelegte Daten »verbogen« (verändert). Eingaben werden ins Leere umgeleitet und können nicht verarbeitet werden. Die einzige Lösung einer solchen lockup- Situation ist in der Regel die unbeliebte Drei-Finger-Lösung — der -> reset.

logical device (auch logical directory oder system directory) Ein Element des -> Amiga-DOS, um das Auffinden der Dateien für bestimmte systembezogene Zwecke oder für bestimmte Subsysteme zu erleichtern. Das jeweilige Verzeichnis wird vom System als eine besondere Diskette (volume) oder -> physical device betrachtet. Dies ist an dem abschließenden Doppelpunkt bei der Bezeichnung dieser Systemverzeichnisse zu erkennen.

Amiga-DOS kennt sieben solche logical devices: SYS:, C:, L:, S:, LIBS:, DEVS: und FONTS:. Der Benutzer kann mit dem ASSIGN-Befehl entweder die Zuweisung dieser devices zu bestimmten Verzeichnissen ändern oder andere Verzeichnisse zu logical devices ernennen, um so zum Beispiel einen Suchpfad (-> path) zu vereinfachen.

long word alignment Ein long word ist ein aus 32 Bit bestehender Wert (4 Byte). Die verschiedenen Prozessoren des Amiga benötigen für bestimmte Operationen Daten, die an einer sogenannten Wortgrenze beginnen, das heißt, an einer durch vier teilbaren Speicheradresse. Die Positionierung von Daten an einer solchen Speichergrenze bezeichnet man als »long word aligned« (siehe word alignment).

low pass filter (Tiefpaßfilter) Ein besonderer Schaltkreis des Audio-Subsystems, der niedrige Frequenzen verstärkt und höhere Frequenzen herausfiltert. Der Tiefpaßfilter eines Audiokanals dient in erster Linie der Unterbindung von »aliasing distortion«. Das sind Störfrequenzen, die durch das Zusammenspiel der Samplingfrequenz und der zu erzeugenden Frequenz entstehen. Diese Störfrequenzen treten in zwei Bereichen auf, die sich durch die folgende Berechnung ermitteln lassen: Samplingfrequenz + Tonfrequenz und Samplingfrequenz — Tonfrequenz. Der Frequenzbereich des Filters reicht von 0 bis 7 kHz. Daraus folgt, daß die niedrigste Samplingfrequenz um mindestens 7 kHz höher als die höchste Tonfrequenz sein muß, sollen diese beiden Störfrequenzen nicht entstehen. Anders ausgedrückt, bei einer zu erzeugenden Frequenz bis zu 7 kHz kann diese nur bei einer Samplingfrequenz von 14 bis 28 kHz ohne aliasing distortion erzeugt werden (siehe quantization noise, sound synthesis).

magic (numbers) (Zauberzahlen). Diese besser als Schlüsselzahlen bezeichneten Werte werden von bestimmten Subsystemen überprüft, um festzustellen, ob die angesprochene Datenstruktur zum richtigen Typ gehört. So wird zum Beispiel bei einer .info-Datei überprüft, ob diese zu Beginn die Hexzahl $E310 enthält — die Kennung für Icondaten.

mask 1 (Bitmaske) Eine bestimmte Bitfolge, mit der andere Daten logisch verknüpft werden. Vor allem vom Grafiksubsystem werden solche Masken benutzt. Beispiele:

— »RastPort.Mask« bestimmt, welche -> Bitplanes des Screen mit Daten beschrieben werden dürfen.

— »HitMask« und »MeMask« werden bei der Kollisionsverarbeitung von BOBs und Sprites benötigt.

— »BltPattern« bestimmen bei der Verknüpfung der Maske mit den Daten der Bitmap durch den Blitter den aktuellem Bildschirminhalt.

— »ImageShadow« enthält alle gesetzten Pixel eines BOBs oder Sprites und kann mit HitMask identisch sein. Sie dient als Ausgangspunkt für eine 16-Bit-Maske, mit der Kollisionen mit Windowgrenzen schneller festgestellt werden können.

— »PlanePick« und »PlaneOnOff« sind eigentlich Systemvariablen, funktionieren aber wie Masken:

Mit »PlanePick« wird bestimmt, welche Bitplanes aktiv sein sollen und damit die Anzahl der verfügbaren Farben. Verschiedene Sprites mit unterschiedlichem PlanePick-Wert aber gleicher Gestalt können unterschiedlichen Farben besitzen. »PlaneOnOff« bestimmt,

was mit den von »PlanePick« nicht aktivierten Bitplanes geschehen soll. Dies hat wiederum Auswirkungen auf die Farbwahl, denn die mit »PlaneOnOff« bestimmten Bitplanes werden so behandelt, als wären alle Pixel der Bitplane gesetzt.

Mask 2 (Eingabemaske) Das »Formular« für die Dateneingabe. In der Regel für die Dateiverwaltung genutzter Bildschirmaufbau, bei dem die einzugebenden Felder durch Striche oder Rechtecke markiert sind.

Mathffp.library Die Routinen für die Fließkommaarithmetik und die Grundrechenarten. In Assemblerprogramme müssen diese Routinen durch die Angabe von »amiga.lib« beim Linken eingebunden werden. Die Library enthält auch die Umwandlungsroutinen für ASCII nach FFP, FFP nach ASCII und FFP nach BCD, die mit mathlink_lib.lib in C (nicht in Assembler) eingebunden werden können.

MathIeeeDoubBas.library Programmroutinen für Berechnungen mit Fließkommazahlen doppelter Genauigkeit.

MathTrans.library Die Routinen für die »transzendentale« Fließkomma-Mathematik. Transzendentale Funktionen sind die trigonometrischen Funktionen (Sinus, Cosinus, ...). In Assemblerprogramme müssen die Routinen mit »mathlink.lib« eingebunden werden. Um die MathTrans.library zu benutzen, muß vom Programm zuerst die »mathffp.library« geöffnet werden.

memory (der Speicher) Der RAM-Speicher im Computer wird allgemein »primary memory« (Primärspeicher) genannt, während die externen Speichermedien (Floppy, Festplatte, Kassette, CD-Disk, ...) als »secondary Memory« (Sekundärspeicher) bezeichnet werden. Die beiden Speichertypen unterscheiden sich in der Regel dadurch, daß der Primärspeicher flüchtig (volatile) ist. Das bedeutet, daß die dort enthaltenen Daten beim Ausschalten des Gerätes oder einem Reset verloren gehen. Die Daten des Sekundärspeicher sind nicht flüchtig ((semi)permanent,non volatile) und mittels eines Datenträgers meist von einem Gerät zum anderen zu übertragen.

memory access (Speicherzugriff) Man unterscheidet den »memory access cycle« als die Folge der erforderlichen Operationen, um einen Datenwert aus dem Speicher zu lesen oder in den Speicher zu schreiben und die »memory access time« als Dauer dieser Operationsfolge (siehe system timing frequencies).

memory management (Speicherverwaltung) Da der Amiga ein multitaskingfähiger Computer ist, müssen die Codesegmente überall im Speicher lauffähig sein. Das bedeutet, daß der Programmcode frei verschiebbar (relokierbar) sein muß. Die Stelle (Adresse), an der ein Code- oder Daten-Segment im Speicher plaziert wird, bestimmt -> Exec anhand einer -> free list und der Tasktabelle. Eine typische Situation sieht (auf einem 256 KByte-Amiga) folgendermaßen aus (Anfangsadressen der Codesegmente):

$000EA8 file system
$0027D2 input.device
$004F92 trackdisk.device
$00A8D0 CON:
$02069E printer.device
$015EA8 Workbench
$022F98 CON:

Anders als bei anderen Systemen wird beim Amiga keine »memory management unit« (Speicherverwaltungsbaustein), sondern eine Reihe von Systemlisten verwendet. Aus diesem Grund kann ein an falscher Adresse abgelegter Wert leicht das ganze System durcheinander bringen. Ein blindes »POKEn« im Speicher ist eine ziemlich sichere Methode, den Guru hervorzulocken (siehe fragmentation, heap, scatter load).

memory map Die Belegung des von der CPU adressierbaren Speicherbereichs. Die CPU im Amiga, der MC 68000, kann mit seinem 24-Bit-Adreßbus insgesamt 16 MByte (16.777.216 Byte) adressieren. Bei der Aufteilung dieses Bereiches haben die Amiga-Entwickler sich anscheinend an das von den 8-Bit-Computern bekannte Modell mit acht Speicherblöcken, diesmal mit je 2 MByte statt 8 KByte, gehalten. Zwischen dem Erscheinen des A1000 und der neuen Modelle A500/A2000 haben die Entwickler auch diese Belegung anders gestaltet. Dadurch können die neuen Modelle 9,5 statt 8,5 MByte Speicher nutzen. Die derzeit aktuelle Speicherbelegung sieht folgendermaßen aus (alle Adressen in Hexwerten):

BLOCK 0:
000000 bis 03FFFF:
Die eingebauten 256 KByte des A1000.
040000 bis 07FFFF:
Die 256 KByte Erweiterung des A1000.
Diese 512 KByte befinden sich beim A500 und A2000 ebenfalls an dieser Stelle.
080000 bis 1FFFFF:
reserviert

BLOCK 1-3:
200000 bis 7FFFFF:
Erweiterungen mit 3 x 2 MByte

BLOCK 4
800000 bis 9FFFFF:
Erweiterung von 1 x 2 MByte oder Sidecar beziehungsweise Janus oder sonstige Hardwareerweiterungen, die mehr als 4 KByte Speicher für die Einbindung in das System benötigen (zum Beispiel die Festplatte).
Bei Sidecar beziehungsweise Janus wird der Bereich folgendermaßen aufgeteilt:
800000 Speicher für Bytewerte
820000 Speicher für Wortwerte
840000 Grafikspeicher (Langwortwerte)
860000 i/o Register
880000 PC ROM
8C8000 PC Festplatte
8E0000 Schnittstelleninitialisierung

BLOCK 5:
A00000 bis BFFFFF:
Nur in den obersten 4 KByte von den Registern der beiden 8520 i/o-Chips belegt

BLOCK 6:
C00000 bis CFFFFF:
Ein nachträglich für den Speicherausbau freigegebener Bereich von 1 MByte. Die 512K-Karte des A500 und A2000 belegt den Bereich von C00000 bis C7FFFF.
D00000 bis D7FFFF:
Für -> auto.config-Erweiterung nutzbar. Bei der Erweiterung ist darauf zu achten, daß dieser Bereich eventuell von Hardwarezusätzen belegt werden kann.
ANMERKUNG: die auto.config-Routinen füllen anscheinend den nicht ausgebauten Speicherbereich von der höchsten Adresse abwärts in Blöcken zu je 64 KByte, das heißt, eine fortlaufende Erweiterung einer RAM-Karte muß in 64 KByte-Schritten erfolgen. Bei vorhandenem Fast-RAM wird Amiga-DOS in diesen Bereich geladen und das knappe Chip-RAM kann für andere Zwecke genutzt werden.
D80000 bis DFFFFF:
Nur in den obersten 4 KByte von den Registern der Sonderchips Agnus, Denise und Paula belegt.

BLOCK 7:
E00000 bis E7FFFF:
Reserviert E80000 bis EFFFFF:
Reserviert für 128 x 4 KByte »external expansion decoder slots«.
F00000 bis F7FFFF:
Für das Betriebssystem reserviert.
F80000 bis FFFFFF:
ROM Kernel:
A1000: 64 KByte Boot-ROM + 192 KByte WCS (Kickstart)
A500/A2000: 256 KByte System-ROM Version 1.2

menu Die »Speisekarte« der Funktionen eines Programms. Diese können mit der Maus ausgewählt werden. Ein Menü wird immer einem bestimmten -> Window (und damit einem Programm) oder einem -> Screen zugewiesen, obwohl die verschiedenen Funktionsleisten immer in der Titelleiste des entsprechenden Screens erscheinen. Die erscheinende Menüleiste (menu strip) gehört immer zum Programm des aktuellen Windows beziehungsweise Screens.

Beim Amiga werden sogenannte »drop down«- oder »roll down«-Menüs benutzt. Diese erscheinen (»fallen herunter«), wenn der aktive Punkt des Zeigers bei gedrückter Maustaste in der Menüleiste an einer Stelle steht, an der sich das Menü befindet. Eine andere Technik verwenden die sogenannten »pull down«-Menüs, die für eine Funktionsanwahl tatsächlich mit der Maus — wie eine Art Rollo — nach unten gezogen werden müssen.

Ein Menü enthält eine Liste von Steuerfunktionen (menu items), die ihrerseits auch untergeordnete Funktionen (subitems) enthalten können. Das Betriebssystem erlaubt

maximal 32 Menüs mit jeweils maximal 32 Funktionen. Für jede Menüfunktion kann der Programmierer wiederum maximal 63 Nebenfunktionen definieren. Bis zu 32 Menüoptionen können auch als -> mutual exclude Optionen deklariert oder mit -> keyboard shortcuts versehen werden. Nicht auswählbare Optionen werden in »Geisterschrift« (-> ghosting) angezeigt.

Menüfunktionen können »attributes« oder »actions« ändern oder auslösen. Eine Attributfunktion wird in der Regel durch das -> checkmark identifiziert und bleibt gewählt, bis das jeweilige Attribut wieder deaktiviert wird (Beispiel: Textattribute fett, unterstrichen, kursiv bei Textverarbeitung). Eine Aktionsfunktion hat nur einmalige Gültigkeit und muß jedesmal ausdrücklich neu gewählt werden. Solche Funktionen werden auch als »hit select item« bezeichnet (Beispiel: die Funktionen Cut, Copy, Paste von Amiga-Basic).

Es gibt auch ein sogenanntes Doppelklickmenü, welches aber eigentlich eine besondere Form von -> Requester ist. Anders als beim Requester kann ein Menü verlassen werden, ohne daß eine Eingabe (= Funktionswahl) erfolgen muß.

Die Beispiele zu Menüs im -> RKM sind nicht immer genau. Für die richtigen Strukturen sollte man besser die entsprechenden -> include files heranziehen.

message (Mitteilung, systeminterne Kommunikation zwischen einzelnen -> tasks, processes) Die Grundlage dieses Kommunikationssystems sind die messages selbst und die jeweiligen Sender beziehungsweise Empfänger: die message ports. Das ganze Subsystem wird von -> Exec verwaltet. Ein »message port« ist eine Exec-Datenstruktur, deren Speicheradresse Sender und Empfänger bekannt ist. Als »public port« ist dieser mit einem Namen versehene »Briefkasten« für alle tasks erreichbar. Diese Datenstruktur wird als -> linked list geführt. Ankommende Mitteilungen werden hinten an die Liste angehängt, die Ankunft wird durch ein -> signal oder semaphore mitgeteilt. Dem empfangenen task/process obliegt es, diese Mitteilung(en) abzuholen und zu verarbeiten.

Die Nachricht im message port des Empfängers besteht allerdings nur aus einer Speicheradresse. Diese Adresse verweist auf einen Speicherbereich des Senders, in dem die eigentliche Nachricht (message) zu finden ist. Eine message ist somit die Erlaubnis des Senders an den Empfänger, auf diese Daten zugreifen zu dürfen.

Nach der Verarbeitung der Daten schickt der Empfänger die message (die Speicheradresse) an den Sender zurück. In diesem Fall wird das empfangende message port »reply port« genannt. Dadurch wird der vorübergehend beanspruchte Speicherbereich dem »Eigentümer« wieder zurückgegeben. Erfolgt diese Rückgabe nicht, geht der Speicherbereich für den Sender verloren. Wiederholte Nichtrückgabe der Nachricht kann zu einem Systemabsturz (out of memory oder stack overflow) führen.

Das message-Subsystem steuert auch den Zugang zu den -> resources. Die laufenden Prozesse synchronisieren auf diese Weise — mit Hilfe von Exec — ihre Abläufe und Ressourcezugriffe.

MetaComCo Englische Softwarefirma, die — der Legende nach — erst dann von Commodore beauftragt wurde, nachdem eine amerikanische Firma fast zwei Jahre lang vergeblich versucht hatte, ein multitasking-fähiges Betriebssystem zustande zu bringen. MetaComCo soll innerhalb von drei Monaten nicht nur -> Amiga-DOS, sondern auch das ABasiC geliefert haben. Obwohl ABasiC von Microsofts Amiga-Basic abgelöst wurde, hat sich MetaComCo, ähnlich wie -> E.A. beim -> IFF als führender Entwickler des DOS und -> CLI etabliert. Dies nicht zuletzt deshalb, weil diese Teile des Betriebssystems weitgehend in der in Amerika selten benutzten Sprache -> BCPL geschrieben worden sind.

MFM -> pulse code modulation

MIDI (Musical Instrument Digital Interface) Ein normiertes Übertragungsprotokoll für elektronisch gesteuerte Instrumente (Beispiel: Synthesizer). Der Amiga kann ein solches Protokoll unterstützen, ist aber nicht serienmäßig mit der erforderlichen Schnittstelle ausgerüstet. Entsprechende Nachrüstungen sind aber relativ preiswert. Fast alle bisher erschienene Musikprogramme für den Amiga enthalten eine MIDI-Funktion.

minterm variable Bei den Logikoperationen des -> Blitter eine von acht möglichen logischen Kombinationen der Datenbits aus den drei verschiedenen Quellen (den drei »Eingabekanälen« des Blitter). Daraus ergeben sich die insgesamt 256 Logikoperation des Blitter. Die gewählten minterms werden in das Register BLTCON0 des Blitter geschrieben (siehe Hardware Reference Manual Seite 171 bis 177).

MMU -> memory management unit

mnemonic In der Regel bei der Assemblersprache verwendete Bezeichnung für einenen Maschinensprachebefehl (MOVE, ADD ...). Statt der schlecht merkbaren binären Verschlüsselung von Befehlen werden symbolische Formen verwendet, die sich der natürlichen Sprache möglichst annähern und damit die Merkfähigkeit verbessern.

modem (MOdulator-DEModulator) Ein für die Datenfernübertragung genutztes Verbindungsgerät zwischen Computer und Telefonnetz, das für den Sendevorgang digitale Signale des Computers in analoge Telefonsignale, und diese umgekehrt beim Empfang wieder in digitale Signale umwandelt.

Ursprünglich sollte der Amiga — damals unter dem Namen »Lorraine« bekannt — mit einem eingebauten Modem ausgerüstet werden. Nach Übernahme der Firma Amiga durch Commodore wurde das Modem aber aus der Planung genommen.

modulo Der Rest bei einer arithmetischen Division. Beim Amiga werden Modulo-Werte insbesonders für den -> Blitter beim horizontalen Verschieben des Bildes benötigt. Der Wert des Modulos ist die Anzahl der Datenbytes zwischem dem letzten Byte einer Bildschirmzeile und dem ersten darzustellenden Byte der folgenden Zeile. Beim normalen Bildaufbau spielt dieser Wert eine wesentliche Rolle, denn mit dessen Hilfe werden die linear gespeicherten Daten einer -> Bitplane in das rechteckige Format für die Bildschirmdarstellung umgewandelt. Dasselbe Verfahren wird auch für die Verschiebung der -> BOBs benötigt.

motion control Die Geschwindigkeits- und Beschleunigungssteuerung von -> GELs. Als eingebaute Animationsfunktionen bewegen die zuständigen Systemroutinen des Amiga BOBs und Sprites gemäß eingestellter Werte für Grundgeschwindigkeit und gegebenfalls Beschleunigung. Dieses Verfahren wird auch bei den Amiga-Basic-Befehlen OBJECT.AX/.AY und OBJECT. VX/.VY sowie den entsprechenden Funktionen angewendet. Die Geschwindigkeitsangabe wird als binäre Festkommazahl mit dem Dezimalpunkt zwischen Bit 6 und Bit 7 interpretiert. Das bedeutet, daß sich die Geschwindigkeit auf $^1/_{64}$ einer Zeiteinheit (in diesem Fall 1 Sekunde) genau regulieren läßt. Alle Bewegungen werden als Anzahl Pixel pro Zeiteinheit ausgeführt. Die Beschleunigungsangabe wird jede Sekunde zu der Grundgeschwindigkeit addiert. Vorsicht ist bei der Beschleunigungsangabe geboten, denn bei hoher Beschleunigung erreichen die GELs sehr schnell die »Fluchtgeschwindigkeit« und verschwinden auf Nimmerwiedersehen, weil die Windowgrenze zwischen zwei Positionen liegt und das System keine Zeit hat, eine Kollision zu registrieren, geschweige denn zu verarbeiten (siehe animation, sequenced drawing).

mount (in Stellung bringen, aufstellen) Auf den Amiga bezogen: Ein externes Speichergerät in das Betriebssystem einbinden (siehe auto. config). Erst unter Version 1.2 aktuell, weil sich die entsprechenden CLI-Befehle (MOUNT, BINDDRIVERS) vorher nicht auf der Workbench befanden.

Der Begriff bezieht sich nicht nur auf Laufwerke jeder Art, sondern auch auf Disketten (siehe INFO vom CLI aus), die auch in das System eingebunden werden; entweder enthält die Diskette ein Verzeichnis, das einem -> logical device zugewiesen wurde oder der Diskpuffer enthält Daten, die zu einer bestimmten Diskette gehören (»dirty track«: die Daten sind eventuell verändert, aber noch noch nicht auf die Diskette zurück geschrieben worden). Ferner kann der Diskpuffer unter Umständen das »root directory« einer Diskette als »current directory« enthalten. In all diesen Fällen werden die Disketten beim CLI-Befehl INFO als »mounted« (verfügbar) gemeldet.

mount list Datenstruktur für die Einbindung eines externen Speichergeräts in das System. Die Einträge in der Datei »mountlist« enthalten Angaben, die vom System beziehungsweise dem zuständigen -> .device benötigt werden, um das neue Gerät in die entsprechende Systemliste einzutragen.

mouse Das Hauptsteuergerät des Amiga unter -> Intuition. Eine Maus ist ein auf den Rücken gedrehter Trackball-Controller mit bis zu drei Bedienungstasten anstelle der Feuerknöpfe. Die Amiga-Maus hat zwei Tasten. Das Betriebssystem kann bei entsprechender Installierung Mausausführungen mit bis zu drei Tasten unterstützen.

In erster Linie steuert die Maus den Mauszeiger (mouse pointer oder window pointer). Mit den beiden Tasten werden entweder Menüoptionen gewählt (rechte Taste = select key) oder irgendeine Manipulation am Bildschirm vorgenommen (Icon, Screen oder Window verschieben) beziehungsweise irgendeine Tätigkeit (Programm starten, requester aktivieren (1.1)) ausgelöst (linke Taste = action key). Den Maustasten sind die beiden Amiga-Tasten zugeordnet (rechts Selektion, links Aktion).

Das Betätigen einer Maustaste wird als »klicken« bezeichnet. Dabei unterscheidet man zwischen »anklicken« (einmalige Betätigung) und »doppelklicken«. Es kann bei manchem Programm aber auch vorkommen, daß man beide Tasten gleichzeitig betätigen muß. Dafür gibt es allerdings noch keine gängige Bezeichnung.

Bei der Wahl einer Menüfunktion genügt es in der Regel, das Symbol mit der Maus anzusteuern und — sobald sich das entsprechende Symbol optisch ändert (-> highlight) — die rechte Taste loszulassen. Ein Sonderfall ist die »Doppelklickfunktion«, mit der ein -> requester abgerufen werden kann.

Die Steuerkugel der Maus ist gegen Dreck und klebrige Substanzen empfindlich. Auf Hochglanzoberflächen und bei zu großem Druck auf das Gehäuse der Maus gleitet die Kugel anstatt zu rollen. Dagegen werden sogenannte »mouse pads« angeboten, aber eine normale Schreibtischunterlage erfüllt denselben Zweck.

Angeschlossen wird die Maus am sogenannten »mouse port« (eigentlich gameport 1). Die Auswertung der Maustätigkeiten erfolgt über das -> input.device und -> Intuition.

FMS-DOS (Microsoft Disc Operating System) Von Microsoft in Seattle/USA entwickeltes Betriebssystem. Ist auf Personal Computern sehr verbreitet und wird daher auch als PC-DOS bezeichnet.

multiprocessing Ursprünglich bedeutete dieser Begriff die Verwendung von mehreren, parallel geschalteten CPUs, um die Datenverarbeitung zu beschleunigen. Neuerdings geht man mehr dazu über, dieses Verfahren (genauer) »parallel processing« zu nennen und den Begriff »multi processing« für die Beschreibung von Systemen zu benutzen, in denen eine CPU durch mehrere untergeordnete Hilfsprozessoren unterstützt wird.

node (Eintrag, Codesegment) Ein Eintrag in einer -> linked list oder ein Codemodul in einem binären -> load file (siehe hunk).

nondestructive readout (NRDO) Bei einigen Registern, die man nur auslesen aber nicht beschreiben kann, führt das Auslesen zum Zurücksetzen des Registerinhalts (destructive readout: -> collision stack von Amiga-Basic). Beim NDRO bleibt der aktuelle Inhalt des Registers unverändert erhalten.

NOTEPAD Der »Notizblock« der -> Workbench.

Von vielen Anwendern des Amiga als Textverarbeitungsersatz verwendet. Er bietet unter anderem eine Seitennumerierung, frei wählbare Zeichensätze, Editieroptionen (cut, copy, paste, find, replace), Blockmarkierung, automatisches »word wrap« (ein Wort, das nicht mehr in die Zeile paßt, wird vollständig in die nächste Zeile übernommen). Um von allen Zeichensätzen Gebrauch machen zu können, muß die Funktion »global fonts« ausgeschaltet werden.

In diesem Sinne ist der Amiga ein Multiprozessorsystem. Operationen, die sonst die CPU bearbeitet hat, werden von Hilfsprozessoren übernommen: Bildschirmprozessor Copper; Animations- und DMA-Prozessor Blitter; Audio- und I/O-Prozessor Paula, Video und I/O-Prozessor Denise und Tastaturprozessor 6500/1. Dazu kommen weitere Hilfsprozessoren bei verschiedenen Hardwareerweiterungen: Festplattencontroller, die Chips der Janusschnittstelle und andere). Alle diese Hilfsprozessoren dienen dazu, die CPU zu entlasten.

Beim A2000 bietet der »processor slot« die Möglichkeit, ein echtes Parallelprozessorsystem zu realisieren. Eine parallele Datenverarbeitung ist auch in Verbindung mit dem bridge board und dem PC-seitigen Prozessor vorstellbar, vor allem, wenn dieser durch einen (x)xx87 Mathematik-Coprozessor unterstützt wird.

multitasking Die Fähigkeit eines Computers, (scheinbar) mehrere Programme gleichzeitig ablaufen zu lassen. Im weiten Sinne des Begriffes wird auch die Fähigkeit, mehrere Benutzer gleichzeitig arbeiten zu lassen, eingeschlossen. Hier ist aber die Bezeichnung »multiuser« (Mehrbenutzersystem) zutreffender.

Das Multitaskingsubsystem des Amiga wird von -> Exec auf der task-Ebene und von -> Amiga-DOS auf der process-Ebene verwaltet. Die Grundlage hierfür ist das -> message-Subsystem, mit dem die verschiedenen Programme synchronisiert werden, und reihum Zugang zu den -> resources erhalten (siehe timesharing).

mutual exclusion Gegenseitige(r) Abschaltung/Ausschluß. Vor allem bei Menüfunktionen gibt es Operationen, die nicht gleichzeitig ausgeführt werden können (Beispiel: LOAD und SAVE). Durch das Setzen einer entsprechenden Flagge (eines Wertes) in der jeweiligen Datenstruktur wird dafür gesorgt, daß die Systemroutinen solche Konflikte automatisch lösen, indem eine bereits gewählte, aber kollidierende, Funktion bei der Wahl der neuen Funktion abgeschaltet wird (siehe toggle).

narrator.device Die Systemroutine für die Umwandlung von Phonemcodes in Wellenformen für die Sprachsynthese. Als Beigabe liefert diese Routine auch Werte für Höhe und Breite des Mundes bei den einzelnen Vokallauten (siehe speech synthesis).

nesting (Verschachtelung) Das Einbetten einer bestimmten Programm- oder Datenstruktur in eine andere Struktur von (in der Regel) demselben Typ. (Beispiel: FOR..NEXT-Schleifen in Basic oder die user -> directories (Unterverzeichnisse) auf einer Diskette.

NIL Ein besonderes -> logical device von -> Amiga-DOS. Es kann nur als Ausgabegerät benutzt werden. Eingaben von NIL: werden stets als Dateiendemarkierung interpretiert. Das Gerät hat den englischen Beinamen »bit bucket« (Abfluß oder Abfalleimer) erhalten. Über NIL: ausgegebene Daten werden einfach »weggespült«. Sinn dieser Einrichtung ist es, den Programmierern eine Möglichkeit zu geben, Programm(abschnitt)e zu testen, ohne daß der Speicher mit Daten vollgestopft wird. So kann in Testläufen der Programmfluß ohne Berücksichtigung der Ergebnisse überprüft werden. Als Nebenaufgabe dient NIL: zur Unterbindung der ungewollten Ausgabe von -> argument templates bei der Benutzung von bestimmten CLI-Befehlen in -> batch files (LoadWB in der -> startup-sequence).

* = Diese Programme sind auf der Programmsammlungs-Diskette erhältlich.

Faszinierende Bilder aus der Juliamenge

Es gibt schon viele Programme zur Darstellung von Mandelbrotmengen (Apfelmännchen). Unbekannter, aber deswegen nicht minder interessant, sind die sogenannten Juliamengen. Auch sie ergeben schöne Bilder.

Wie auch bei den Mandelbrotmengen beruht die Berechnung von Juliamengen (ein Beispiel sehen Sie in Bild 1) auf komplexen Zahlen. Dieses sehr umfangreiche und komplizierte Thema soll hier nicht ausführlich besprochen werden, aber einige wichtige Grundlagen sind es wert, kurz beleuchtet zu werden.

Es handelt sich in beiden Fällen um einfache rückgekoppelte Vorgänge, bei denen eine bestimmte Rechenvorschrift wiederholt durchlaufen wird. Dabei dient das Ergebnis als der Eingangswert für den nächsten Durchlauf. Im Bild 2 sehen Sie diesen Vorgang symbolisch dargestellt.

Starten wir unsere Iteration mit dem willkürlich gewählten Wert z0, so erhalten wir eine Reihe von Werten (z1, z2, z3, ...). Diese Werte sind komplexe Zahlen und besitzen somit einen reellen und einen imaginären Anteil.

Die Formel lautet also:

```
zk+1 = f(zk, c)
```

Lösen wir diese Gleichung nach Real- und Imaginärteil (hier xk und yk) auf, so erhalten wir:

```
xk+1 = f(xk,yk,p)
yk+1 = f(xk,yk,q)
```

Ein Weg ist die Wahl eines festen Punktes (x,y) und die Verfolgung seines Verhaltens während der Iteration für verschiedene Werte des Parameters c(p,q). Stellt man das Ergebnis dann Punkt für Punkt in der p-, q-Ebene dar, so erhält man die Mandelbrotmenge (Apfelmännchen).

Betrachtet man jedoch die x-, y-Ebene bei festen Werten für p und q und läßt die Iteration für den Punkt (x, y) in der x-, y-Ebene darstellen, so erhält man die Juliamenge.

In dem Programm »Julia« werden für ein komplexes Polynom die Nullstellen berechnet. Die Umgebung dieser Nullstellen wird dann in verschiedenen Farben dargestellt.

Der darzustellende Bereich der x-, y-Ebene wird mit den Variablen xmin, xmax, ymin und ymax festgelegt. Die Schrittweiten in X- und Y-Richtung (dx und dy) berechnen sich dann nach den beiden folgenden Formeln:

```
dx=(xmax-xmin)/pa%
dy=(ymax-ymin)/pb%
```

Hierbei sind pa% und pb% die Anzahl der Bildpunkte des Bildschirms in X- beziehungsweise Y-Richtung.

Für jeden Punkt wird nun die Iteration, beginnend bei xmin und ymin bis xmax und ymax, durchgeführt. Die Koordinaten X und Y werden dabei für die Bildschirmkoordinaten i und j folgendermaßen berechnet:

```
x=xmin+dx*i%
y=ymin+dy*j%
```

Bis jetzt ist aber noch nicht entschieden, welche Farbe der Punkt erhalten soll. Aber genau damit kann man das Aussehen des Bildes entscheidend beeinflussen.

In vorliegendem Programm wird (mit leichten Modifikationen) folgender Weg gegangen. Die Funktion besitzt drei Nullstellen, denen wir je einen Farbsatz zuordnen. Da im hochauflösenden Modus 16 Farben zur Verfügung stehen, erhält jede Nullstelle einen Farbsatz von fünf Farben. Die letzte Farbe findet als Hintergrund Verwendung.

Wir iterieren nun k-mal und fragen nach jedem Schritt, ob die errechneten Werte bereits einer der Nullstellen entsprechen. Ist das der Fall, berechnen wir daraus den Farbsatz und die Einzelfarbe. Ansonsten fahren wir mit der Iteration fort.

Zur Bestimmung der Einzelfarbe verknüpfen wir die Zahl der benötigten Iterationen über die Modulo-Funktion mit der Zahl 5. Dadurch erhalten wir immer einen Wert von 0 bis 4, der eine bestimmte Farbe darstellt. Dieses Verfahren stellt nur einen der vielen denkbaren Wege dar. Versuchen Sie ruhig eigene Modifikationen. Anregungen sind zum Beispiel:

— Verschieden große Farbsätze für die Nullstellen
— Einsetzen einer anderen Verknüpfung statt Modulo, zum Beispiel die UND-Verknüpfung
— Andere Abbruchbedingungen für die Iteration, zum Beispiel nicht das exakte Erreichen einer Nullstelle, sondern ein bestimmtes Intervall

Es gibt eine Fülle von Möglichkeiten, die alle zu unterschiedlichen, aber meist faszinierenden Bildern führen.

Ein wichtiger Punkt des Programms kam bis jetzt noch nicht zur Sprache: die Bildsymmetrierung. Dies Funktion wurde aus zwei Gründen eingebaut. Erstens ist das Verhältnis von Höhe zu Breite des Bildes bei jedem Bildschirm verschieden, und zweitens ist es schwierig die Werte xmin, xmax, ymin und ymax so zu wählen, daß sie mit dem Seitenverhältnis des Monitors übereinstimmen.

Messen Sie also anhand eines fertig berechneten Bildes die Breite und Höhe der Darstellung aus. Die so erhaltenen Werte tragen Sie in Zeile 36 in Millimetern ein.

Für den ersten Versuch mit Julia sollten Sie folgende Werte verwenden:

```
xmin = -1.3        Iterationstiefe = 30
xmax = 1.4         R real = 1
ymin = -1.1        R imag = 0
ymax = 1.1         s =0
```

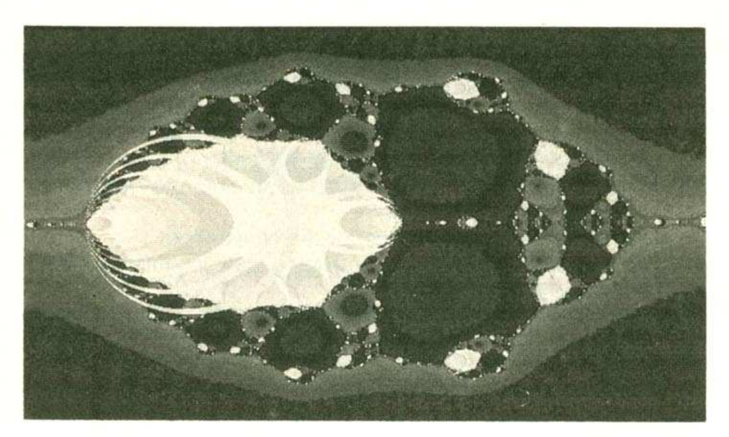


Bild 1. So fantastische Bilder erhalten Sie mit dem Programm »Julia«

Bei Bildsymmetrierung wählen Sie dann 1 oder 2. Durch die Symmetrierung werden die Seitenverhältnisse richtig eingestellt. Es werden je nach Auswahl die reellen oder imaginären Teile konstant gehalten und die anderen entsprechend angepaßt. Mit dem dritten Punkt übernehmen Sie die gewählten Werte, wobei allerdings gestreckte oder gestauchte Bilder entstehen.

Eine Routine zum Abspeichern wurde nicht hinzugefügt, da sie bei Amiga-Basic mitgeliefert wird. Sie finden sie auf der ExtrasDiskette im Ordner BasicDemos. Im Programm »LoadILBM-SaveACBM« ist ein Teil, der Grafiken speichert. Die nötigen Teile zum Laden der Bilder finden Sie im Programm »LoadACBM«.

Die Rechenzeit für ein Bild, bei der angegebenen Funktion liegt je nach Parameterwahl ziemlich hoch. Es gibt zwei Wege, diese zu verkürzen. Damit der Computer nicht unnötig Zeit für den Bildschirmaufbau benötigt, schieben Sie alle Bildschirme ganz nach unten. Dadurch benötigen die Spezialchips des Amiga weniger Zeit für den Bildschirmaufbau, was sich im verwendeten Modus von 640 x 200 Punkten schon sehr deutlich bemerkbar macht. Besser hat es noch derjenige, der einen AC-Basic-Compiler besitzt, da das Programm problemlos übersetzbar ist. Schalten Sie dabei alle Parameter bis auf die Option »R« (Link Run-time) ab. Die compilierte Fassung benötigt dann nur noch $\frac{1}{6}$ der Rechenzeit. Der oben erwähnte Trick mit dem Bildschirm funktioniert natürlich auch.

Auf alle Fälle werden Sie nach der Wartezeit mit fantastischen Bildern aus der Juliamenge belohnt. (Dr. D. Petrausch/rb)

Programmname:	Julia
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2

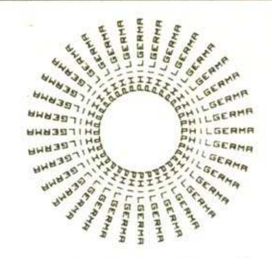
```
Programm : Julia
----------------
 1 QuO REM *************************
 2 Zt  REM ******** FRACTALS ********
 3 71  REM ***** HOCHAUFLOESEND *****
 4 kd  REM ***** R=komplex//s#0 *****
 5 B6  REM *****  p(z)=z**3-1   *****
 6 Vz  REM *************************
 7 c8  REM *** Andreas Petrausch ****
 8 ze  REM ** Dr.Detlef Petrausch ***
 9 ed  REM ****** 09135 - 1076 ******
10 Z3  REM *************************
11 jp  CLS
12 IT  REM ********************** Bilddaten Eingabe **************
       ************
13 hJ  LOCATE 3,19
14 d0  PRINT "Bitte geben Sie die gew"CHR$(252)"nschten Werte ein."

15 nB  LOCATE 5,31
16 KL  INPUT "Bildname : ",nam$
17 1S  LOCATE 8,31
18 jw  INPUT "xmin : ",xmin
19 7Z  LOCATE 9,31
20 dk  INPUT "xmax : ",xmax
21 AJ  LOCATE 10,31
22 u9  INPUT "ymin : ",ymin
23 HR  LOCATE 11,31
24 ox  INPUT "ymax : ",ymax
25 Tf  LOCATE 13,31
26 Yf  INPUT "Iterationstiefe : ",kend%
27 ft  LOCATE 15,31
28 Bi  INPUT "R real : ",u
29 m1  LOCATE 16,31
30 MX  INPUT "R imag : ",v
31 t9  LOCATE 17,31
32 U6  INPUT "s      : ",sc
33 YO  pa%=631 : pb%=383
34 8t  e3%=3 : e6%=6
35 qD  REM ********************** Monitor - Daten *****************
       ************
36 P3  breite=232:hoehe=162
37 QM  vn=breite/hoehe:vq=hoehe/breite
38 aZ  REM ********************* Bildsymmetrierung ****************
       ************
39 g7  sym:
40 CI  CLS
41 Ji  LOCATE 7,30
42 XV  PRINT "Bild symmetrieren :"
43 s8  LOCATE 10,29
44 DB  PRINT "Realteil=const......1"
45 4M  LOCATE 12,29
46 D6  PRINT "Imagteil=const......2"
47 Ga  LOCATE 14,29
48 xa  PRINT "Nein................3"
49 Cx  pa:
50 IH  a$=INKEY$:IF a$="" THEN pa
51 cJ  IF a$="1" THEN
52 cL4     GOTO realteilconst
53 722   ELSEIF a$="2" THEN
54 Ly4     GOTO imagteilconst
55 C82   ELSEIF a$="3" THEN
56 ui4     GOTO label1
57 WF2   ELSE
58 Lf4     GOTO pa
59 tm0 END IF
60 AK  realteilconst:
61 kz  yminn=(ymax+ymin-vq*(xmax-xmin))/2
62 xC  ymaxn=(ymax+ymin+vq*(xmax-xmin))/2
63 r2  ymax=ymaxn:ymin=yminn
64 2q  GOTO label1
65 w0  imagteilconst:
66 Wh  xminn=(xmin+xmax-vn*(ymax-ymin))/2
67 ju  xmaxn=(xmin+xmax+vn*(ymax-ymin))/2
68 qL  xmin=xminn:xmax=xmaxn
69 hw  label1:
70 gm  CLS
71 nL  LOCATE 5,27
72 m3  PRINT "Die neuen Werte lauten :"
73 1c  LOCATE 8,27
74 2C  PRINT "xmin : ",xmin
75 IW  LOCATE 10,27
76 w0  PRINT "xmax : ",xmax
77 Uk  LOCATE 12,27
78 DP  PRINT "ymin : ",ymin
79 gy  LOCATE 14,27
80 7D  PRINT "ymax : ",ymax
81 ix  LOCATE 18,11
82 iv  PRINT "Nach Abschluss der Berechnungen beendigt sich das Pro
       gramm "
83 r8  LOCATE 19,21
84 QX  PRINT "durch dr"CHR$(252)"cken einer beliebigen Taste."
85 dt  LOCATE 22,26
86 WW  PRINT "Weiter mit beliebiger Taste."
87 cx  weiter:
88 A7  a$=INKEY$:IF a$="" THEN weiter
89 z5  CLS
```

Listing. »Julia«, ein Basic-Programm zur Generierung von Bildern aus der Juliamenge. Bitte mit dem Checksummer (Seite 66) eingeben.

```
 90 9E  REM ******************** Screen und Window öffnen **********
        ************
 91 vr  SCREEN 1,640,400,4,4
 92 fP  WINDOW 2,nam$,(0,3)-(631,386),2,1
 93 Id  REM ************************ Farben anlegen ***************
        ************
 94 02  FOR x=0 TO 15
 95 DI3   READ rot,gruen,blau
 96 sE    PALETTE x,rot/16,gruen/16,blau/16
 97 UzO NEXT x
 98 5V  DATA 0,0,0,8,0,0,9,0,0,10,0,0
 99 QW  DATA 11,0,0,12,0,0,0,8,0,0,9,0
100 VK  DATA 0,10,0,0,11,0,0,12,0,0,0,8
101 uL  DATA 0,0,9,0,0,10,0,0,11,0,0,12
102 p5  REM ************************ Bildberechnung ***************
        ************
103 IF  dx=(xmax-xmin)/pa%
104 bS  dy=(ymax-ymin)/pb%
105 hL  FOR j%=0 TO pb%
106 XE7       FOR i%=0 TO pa%
107 1QB         x=xmin+dx*i%
108 Fi          y=ymin+dy*j%
109 HI          FOR k%=0 TO kend%
110 iiF           x2=x*x:y2=y*y:x3=x*x2:y3=y*y2
111 kP            a1=x3-x*y2*e3%-1:a2=x2*e3%-y2*e3%
112 QQ            b1=x2*y*e3%-y3:b2=x*y*e6%+sc
113 hp            a22=a2*a2:b22=b2*b2:ab=a22+b22
114 61            a3=(a1*a2+b1*b2)/ab:a4=(a2*b1-a1*b2)/ab
115 Xk            xx=x-u*a3+v*a4:yy=y-v*a3-u*a4
116 bQ            x=xx:y=yy
117 uJO REM ************************ Farbauswahl *****************
        ************
118 sOF           IF k%>1 THEN
119 OpI             IF (xx=-.5 AND yy=-.8660254) THEN
120 vRL               IF k%=2 THEN
121 OzO                 c%=6
122 ZIL               ELSE
123 dXO                 c%=1
124 wpL               END IF
125 OU                GOTO farbe
126 nkI             ELSEIF (xx=-.5 AND yy=.8660254) THEN
127 2YL               IF k%=2 THEN
128 icO                 c%=1
129 gPL               ELSE
130 OCO                 c%=11
131 3wL               END IF
132 Vb                GOTO farbe
133 hNI             ELSEIF xx>.995 AND xx<1.005 THEN
134 9fL               IF k%=2 THEN
135 THO                 c%=11
136 nWL               ELSE
137 GFO                 c%=6
138 A3L               END IF
139 ci                GOTO farbe
140 C5I             END IF
141 D6F           END IF
142 2PB         NEXT k%
143 qNO farbe:
144 x6A         fa%=(k%-1) MOD 5+c%
145 1fO REM *************************** Plotten *******************
        ************
146 fhE           PSET(i%,j%),fa%
147 OCO REM ****************************************************
        ************
148 4P7       NEXT i%
149 7TO NEXT j%
150 z6  REM ********************** Ende Hauptprogramm *************
        ************
151 a6  BEEP
152 z5  warte:
153 au  a$=INKEY$:IF a$="" THEN warte
154 RN  REM ***************** Screen und Window schliessen ********
        ************
155 HM  SCREEN CLOSE 1
156 iO  WINDOW CLOSE 2
157 HC  END
(C) 1987 M&T
```

Listing. »Julia«, ein Basic-Programm zur Generierung von Bildern aus der Juliamenge. Bitte mit dem Checksummer (Seite 66) eingeben.



AMIGA
COMPUTER-MARKT

Wollen Sie einen gebrauchten Computer verkaufen oder erwerben? Suchen Sie Zubehör? Haben Sie Software anzubieten oder suchen Sie Programme oder Verbindungen? Der COMPUTER-MARKT von »Amiga« bietet allen Computerfans die Gelegenheit, für nur 5,— DM eine private Kleinanzeige mit bis zu 4 Zeilen Text in der Rubrik Ihrer Wahl aufzugeben. Und so kommt Ihre private Kleinanzeige in den COMPUTER-MARKT der **März-Ausgabe** (erscheint am 24. Februar 88): Schicken Sie Ihren Anzeigentext bis zum 21. Januar 88 (Eingangsdatum beim Verlag) an »Amiga«. Später eingehende Aufträge werden in der **April-Ausgabe** (erscheint am 23. März 88) veröffentlicht.

Am besten verwenden Sie dazu die vorbereitete Auftragskarte am Anfang des Heftes. Bitte beachten Sie: Ihr Anzeigentext darf maximal 4 Zeilen mit je 40 Buchstaben betragen. Überweisen Sie den Anzeigenpreis von DM 5,— auf das Postscheckkonto Nr. 14199-803 beim Postscheckamt mit dem Vermerk »Markt & Technik, Amiga« oder schicken Sie uns DM 5,— als Scheck oder in Bargeld. Der Verlag behält sich die Veröffentlichung längerer Texte vor. Kleinanzeigen, die entsprechend gekennzeichnet sind, oder deren Text auf eine gewerbliche Tätigkeit schließen läßt, werden in der Rubrik »Gewerbliche Kleinanzeigen« zum Preis von DM 12,— je Zeile Text veröffentlicht.

Private Kleinanzeigen

Suche: Software

Suche Software für den Amiga 500. Hardware und Literatur wird ebenfalls gesucht. Tel. 07950/635. Bitte nur zwischen 15.00 bis 19.00 Uhr anrufen.

Suche Programme für Amiga 500, kaufe fast alles. Sendet mir eure Angebote an L. Vargel, Scheffelstr. 4, 6901 Mauer

Student sucht Software jeglicher Art für den Amiga 500. Zahle bis 5,— DM/Disk. Angebotsliste wäre gut. Bernhard Groeden, Hamburg, 040/507107/15

Suche Tauschpartner für A500 Software (bevorzug. Raum I.-0). Liste gegen Freiumschlag. Lukas Zimmer, Königsbergerstr. 4, 6580 Idar-Oberstein

Suche PD Soft. Nehme günstigstes Angebot. Verk.: 64er, Floppy, Drucker, Final Cartridge, suche Anleitungen aller Art. Suche origin. Soft bis 15,— DM. Tel.: 09761/1074 Wolfgang

Kontakt zu Amiga-Fans zwecks Erfahrungs- und Softwaretausch gesucht. Telefon 0203/730247 (bitte in der Zeit von 17-20 Uhr anrufen, danke).

Ich suche deutsche Anleitung für Aztec und Lattice C Compiler. Und sonstige Software (evtl. Anleitung)
M. Ludwig, Am Meilwald 28, 8520 Erlangen

Hallo! Suche Software aller Art für A500. Schickt einfach eure Liste + Preisvorstellung an: Thomas Henritzi, Emil Dörlestr. 136, 7834 Herbolzheim oder: Tel. 07643/5383

Amiga-Einsteiger sucht Software.
Insbesondere Lattice C, Deluxe Paint, Deluxe Video und P.D.
H. Köhn, Wrangelstr. 95, 2000 Hamburg-20

Suche Software für Amiga 500
schreibt an Michael Höschle
Flandernstr. 10, 7407 Rottenburg 1

MPS 802 an Amiga? Wer kann helfen?
Software gesucht!
Peer Oldendorf, Schubertstr. 46
2800 Bremen 1, Tel. 0421/346781

Suche Software aller Art
Schreibt an:
Jürgen Schädlich, Hauptstr. 40
3420 Herzberg

Einsteiger Amiga 500 sucht billige Software. Anwendersoft, Grafik, Spiele. Wenn günstig auch Speichererweiterung mit Uhr an: M. Altmeyer, Bochumerstr. 17, 1 Berlin 21

AMIGA 1000 sucht Software und Kontakte! Schickt Listen oder Disks an Ralf Hagemann, Am Thing 2, 2110 Buchholz
Bitte nur günstige Angebote (Spiele, Anw.)

Wer verkauft mir seine alten Originale zu günstigen Preis? Suche u.a.: Faery Tale, S.D.I., Uninvited (auch Utilities) Tel.: 05204/8196

★ STOP ★ AMIGA 500 ★ STOP ★
Amiga-Beginner sucht Amiga-soft. Liste mit Preisvorst.: M. Böhling, Rhabanusstr. 14
★ Dringend ★ 6500 Mainz 1 ★ Dringend ★

Suche für Amiga 500 Software jeglicher Art, wenn möglich mit Anleitung.
Gerhard Thumser, Kantstr. 16,
6094 Bischofsheim

Suche jede Menge Software für A500!
Liste an: Harald Retter, Meisenstr. 1,
6078 Neu-Isenburg 2

→ → → SUCHE TAUSCHPARTNER ← ← ←
Tausche allerneueste Software (nur USA), meldet euch!!! An A. Hoehn an Schwalbenhaag 3, 7730 Villingen > keine Anfänger <

★ ★ ★ ★ ! Deluxe Paint 2 gesucht ! ★ ★ ★ ★
★ Möglichst günstig, nur Original. ★
Falco Kirschbein, Munzinger Str. 3,
7812 Bad Krozingen (nur schriftlich)

Ich suche Tauschpartner für AMIGA-SOFT. Suche außerdem **Copyright-Rechte** von guten **Spielen** und **Anwendern**. Zahle **gut**. Bitte melden: **05103/7507**

● ● ● SUCHE ● ● ●
günstig Spiele SOFTWARE für Amiga 1000
D. Linscheid, Brucknerstr. 3, 5000 Köln 41

Aufbau einer dt. PD-Sammlung. Eingesandte Disk m. Copyrighterklärung = 1 x PD-Disk
Richard Englert, Flößaustr. 160, 8510 Fürth, Tel. 0911/708255

★ Suche orig. Software für Amiga 2000 ★
Lattice CProf, Zorland C2.0, Aztec CCOM.V3.4a, Modula IICOM, Pascal Prof, usw. Angebote an Peter Diel, Siegfriedstr. 15, 6840 Lampertheim 5

Suche Software für meinen Amiga 500!!!
Wie Aegis Draw Pius; Deluxe Print, mit dt. Anl., evtl. auch Tausch möglich.
Willian Vent, Tel. 02043/44087

Suche Software u. Literatur für A500, besonders Strategie, Compiler, Simulationenprogramme, Angebote u. Listen an: Lils Stieglitz, Gelenker-Weg 41, 2434 Grube — Zahle gut

Amiga-Neuling sucht Software für Amiga 500 — Spiele u. Textverarb. Liste an: Michael Schmidt, Am Kühlen Brünnchen 41, 6623 Altenkessel — Tel. 06898/87408

Suche gute Spiele für AMIGA 500. Schickt Eure Listen an Frank Hesselbrock, Mozartstr. 14, 4690 Herne 2

Suche ProWrite und Calligrapher bis je 130 DM Boards Tale bis 40 DM. Georg Seitz, Schwere-Reiter-Str. 35, Geb. 41, 8000 München 40

Suche Navigationsprogramm f. Amiga 1000 zum errechnen/drucken v. Kursen, Flugzeiten, Entfernungen z.B. als Durchführungsplan-FLLOG, H. Schell, Hofwiesen 20, 3062 Bückeburg

Suche Anleitungen von Profi-Software (Originale + gute Fotokopien) spez. Superbase, Draw-Plus u.a. TH. Schäfer, 0271/51486 abds. zahle gut

Suche Software für Amiga
Zahle bis 10 DM pro Disk. Kaufe auch Spiele sendet bitte eine Liste: Thorsten Telge, Wiedetwiete 31b, 2 Wedel Tel. 04103/6665

★ WANTED: Swapfriends für ★
★ neueste Amiga-Software! Call: ★
★ 05922/3960 Andrej 05924/1602 Holger ★
★ jetzt anrufen! Suchen Billigdisks! ★

Suche freundliche AMIGA-Besitzer im Raum Reutlingen/Stuttgart/Nürtingen/Tübingen für Erfahrungs- und Softwaretausch. Tel.: 07127/57974. Nur keine Hemmungen!!!!!!!!!!

C-COMPILER Lattice/Atztek o.a. sowie GRAPHIK-SOFTWARE, DTP u.a. gesucht! Alexander Willbränder, Heckenerstr. 16, D-5469 Windhagen ★ Tel. 02645/4424

Welcher Hobbyastronom hat das Basic-Programm »Planetarium« aus dem Buch »Sterne im Computer« für den Amiga 500 lauffähig gemacht? Tel. 06186/7268

Suche Softwaretauschpartner, Games und Midi-Software für Synthesizer
suche günstig Sidecar, 512 K
0211/751438 abends

AMIGA-Einsteiger sucht günstige Original-Programme aller Art! Bin auch an Hardware interessiert. Adr.: G. Kern, Danziger Str. 14, 7107 Nordheim/Tel.: 07133/15794

Suche Programme aller Art für Amiga 500 vor allem Zing(Keys) und andere Profiware. Schickt eure Listen an H. Folger, Am Seifchen 25, 5920 Bad Berleburg

Suche Assembler und C-Compiler mit Anleitung!?! Wer kopiert mir sie ???
Schickt an H. Folger, Am Seifchen 25,
5920 Bad Berleburg Nicht warten !!!!!

Amiga 500: Ein-/Umsteiger sucht Tauschpartner. Suche auch Original- und Freesoftware aller Art. Programme vorhanden. S. Bremer, Vigölensoll 14, 2430 Neustadt

Amiga Einsteiger sucht Software.
Angebote bitte an B. Jürgensen, Holmberg 4, 2398 Harrislee.

Achtung:

Wir machen unsere Inserenten darauf aufmerksam, daß das **Angebot**, der **Verkauf** oder die **Verbreitung** von urheberrechtlich geschützter Software nur für Originalprogramme erlaubt ist.

Das Herstellen, Anbieten, Verkaufen und Verbreiten von **»Raubkopien«** verstößt gegen das Urheberrechtsgesetz und kann straf- und zivilrechtlich verfolgt werden. Bei Verstößen muß mit Anwalts- und Gerichtskosten von über DM 1 000,— gerechnet werden.

Originalprogramme sind am Copyright-Hinweis und am Originalaufkleber des Datenträgers (Diskette oder Kassette) zu erkennen und normalerweise originalverpackt. Mit dem Kauf von Raubkopien erwirbt der Käufer auch kein Nutzungsrecht und geht das Risiko einer jederzeitigen Beschlagnahmung ein.

Wir bitten unsere Leser in deren eigenem Interesse, Raubkopien von Original-Software weder anzubieten, zu verkaufen noch zu verbreiten. Erziehungsberechtigte haften für ihre Kinder.

Der Verlag wird in Zukunft keine Anzeigen mehr veröffentlichen, die darauf schließen lassen, daß Raubkopien angeboten werden.

Ausland

Suche billige Tauschpartner für den ■ ■ ★ ★ AMIGA ★ ★ (500) Listen an: Roland Schroeder, 19 Quartier Wendel, 3881 Schifflingen, Luxemburg. Schnell!!!

Habe und suche Software, Hardware für den Amiga. Yuval Tempelman, Parkstr. 14, CH-3800 Matten. Tel. (0041) 036/223177, Greetings to all my swap-friends BYE

I'm looking for Software (only new)
write to! Thomas M.
Tsamadou 4
10683 Athina/Greece

Amigafreak sucht Software für den Amiga 1000. Listen an Oliver Frei, Seeblickstr. 15, 8274 Tägerwilen (Schweiz)

■■■■■■ÖSTERREICH■■■■■■
Amiga-Tauschpartner gesucht! Zuschriften bitte an: Johann GRILL/Postfach 277,
1080 Wien ● ● ● ● ● ● ● ● 100% Antwort

★ ★ ★ ★ ★ AMIGA-CH-AMIGA ★ ★ ★ ★ ★
Suche neueste Software! (auch C64)
Contact me: Sandro Kluser, Feldhofstr. 30, 9463 Oberriet, Tel. 071/781790 (ab 18.30)

★ A-500 ★ Tausch ★ ★ ★ Tausch ★ A-500 ★
Suche Tauschpartner in ganzer Welt. Bitte schicke Liste oder Disk zu: Hakan Sundell, Gränsvägen 8, S-663 00 Skoghall, Schweden

★ ★ Switzerland Amiga C-64 Switzerland ★ ★
★ ★ Suche Tauschpartner für Amiga + C-64 Soft! Listen an: Ivan Basso, Postfach 165, CH-7430 Thusis, Tel.: 081/812543

Suche Kontakte zu Amiga- und C 128-Usern wegen Programmtausch, neueste Software ist vorhanden. Hardwareerw. gesucht.
Amiga Postfach 45 A-5020 Salzburg

Suche/tausche neue Software.
Auch PD; Weekend (ab Fr. 19 h) **Call Darth Vader. Tel. Austria = 0043/022312850. Bis bald!**

SCHWEIZ (u.A.)! Suche TAUSCHPARTNER für neue **Software** und Computer**erfahrungen** (C-Programmieren). Matthias LOCHER, Mattenweg 18, CH-4148 Pfeffingen/BL

Habe/Suche neue Amiga-Software
Schreibe oder schicke Liste an:
Ton Machielsen, Vermeerstr. 24,
5301 VE Zaltbommel Holland

Zahle 10 DM für die neuesten Amigaprogramme. Habe z.B. Karatekid 2. Schicken Sie Ihre Disks an: Marsel Sippekamp, Ylsterveste 68, 3432 RT Nieuwegein, Holland

Biete an: Software

Amiga - Original
Superbase (deutsch) Datenbanksystem für Amiga 500, 1000, 2000 zu verkaufen 200,- DM oder Tausch gegen andere Programme Tel. 06103/84069

■ ■ ■ ■ Amiga Software ■ ■ ■ ■
02241/314730 02241/314730
■■■■■■■■■■■■■■■■■■■■

Verk. MCC-Pascal (Orig.) mit Anl. etc.
neuw. 200,— DM VB. Suche Tauschpartner im Kreis Aschaffenburg-Alsenau (Prgs. Tips etc.) ab 18.00 Uhr Tel. 06023/4973. Brandau

Private Kleinanzeigen

Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen

■■■■ Amiga is it!! ■■■■
Suche Tauschpartner für Software, Tips & Tricks, bes. Musik + Anwendungen
Matthias Brixel, Johannisstr. 82, 5000 Köln

Habe 350 Public Disc für Amiga. Würde sie gerne verkaufen, Liste bekommt ihr von Data-World, Postfach 1151, 5067 Kürten (Einzelpreis 6 DM pro Disc)

For The newest Amiga-Stuff write to:
Chr. Schranz
Burscheiderweg 5
1000 Berlin 20 West-Germany

Amiga Anfänger sucht zuverlässigen Tauschpartner, habe hauptsächlich nur ältere Programme.
Tel. 05327/2079 v. 14 bis 18 Uhr (Marc)

Amigasoft immer neu ... Du auch?
call 040/214271

Verkaufe Software + Anleitungen aller Art. Bei guten Preisen. Call 040/214271

Verkaufe VideoScape 3D (dtsch. Anleitung), Preis VHB 280,— DM. Suche Aegis Draw Plus, Logistix (mit Anleitung)!
Tel. 0521/109359 (nach 17 Uhr)

Verkaufe: C für Einsteiger (Data Becker) für DM 25,—, Public Domain Fish & Auge für DM 5/Disk. Peter Kratzer, Grüntenweg 1, 8951 Görisried

★ ★ ★ **TOP-STUFF für AMIGA** ★ ★ ★
contact me: 02204/68231
only 17-21 h OK?
where ave the users in GL/K?????

!! Original-programme mit Handbüchern! UBM-Text, Printmaster, Power-Windows, suche 5¼" Laufwerk 40/80 Umsch/2 MB Golembox/CADPr — Evt. Tausch 06589/1267 nach 18.00 h

PD-Soft für alle Amigas — z.B. Diashows, Diskmonitoren etc. Info kostenlos bei: Th. Wirz, Höhenweg 98, 53 Bonn 1

Verkaufe: MCC-Pascal. ISO-Standard: 138 DM.
Verkaufe: Space Fight; Hindbreaker; Demolition. Einzeln: 18 DM. Zusammen 48 DM (alles Originale) Tel.: 02181/3525

Suche zuverl. Tauschpartner für Amiga-Software. Listen bitte an: Ralf Straub, Taläckerstr. 9, 7104 Obersulm

■■■■■ ! Hello Freaks ! ■■■■■
Suche Tauschpartner für topaktuelle Soft.
■■■■■■■■■■■■■■■■■■■■■■■■
P. Saladin, Vogelsangstr. 7, CH-8180 Bulach

■ Software ■ DPaint1 75,— ★ DVideo 75,— ★ Quiwi 35,— ★ Seven Cities of Gold 35,— ★ Marble Madness 35,— ★ Alles zus. 200,— ★ ab 19 Uhr 04832/7519 (Gunnar verlangen)

★ ★ ★ PUBLIC DOMAIN ★ ★ ★
Tausche PD-Soft! Über 300 Disks mit PD vorhanden! ■ R. Wolf, Tel.: 02541/2874, Deipe Stegge 187, 4420 Coesfeld ■

■■■■■■■■■■■■■■■■■■■■■■■■
■■ ! Tausche neueste Topsoftware ! ■■
■ at once 05103/7518 & nothing else! ■
■■■■■■■■■■■■■■■■■■■■■■■■

Erstelle Anwender-Programme. R. Fischer, Fischergasse 15a, 8898 Schrobenhausen, Tel. 08252/6519 ab 19.00 Uhr

Verkaufe orig. City-Desk, Desktop-Publishing-Programm für DM 180,— Verkauf per NN bei: Patrick Dirr, Breisgaustr. 8, 7830 Emmendingen 14

Biete neueste Amiga-Software (Only News)
Hans-Peter Gökcek
Redwitzstr. 32-34
5000 Köln 41

Habe, suche, tausche Software für Amiga und IBM PC. Schreibt an Frowin Lutz, Kugelangergasse 1, A-6060 Hall in Tirol oder ruft an bei 05223/2191

Verkaufe TRUE-BASIC Interpreter mit Anleitung für DM 180
Tel.: 089/7002858 ab 18.00 Uhr (Frank)

★ ★ ★ ★ Amiga Grafik u. DTP Freaks ★ ★ ★ ★
Umfangreiche Grafik (1000 Bilder) u. Zeichensatz (150) Sammlung. Info gegen Rückporto. R. Hobbold, Gleisstr. 14, 43 Essen 11

Verkaufe: Aegis Sonix = 100 DM, Marble Madness = 20 DM, Starglider = 40 DM, Def. of Crown = 40 DM, Space Battle, Challenger je 10 DM, Two on Two 45 DM/**Alles Originale / Tel. 04174/3234 !!!!**

Verkaufe Vizawrite Amiga. VB 160 DM. Michael Heer, Tel.: 02761/4777 ab 16.00 Uhr.

Ausland

Habe und suche Software für den Amiga. Yuval Tempelman, Parkstr. 14, CH-3800 Matten. (0041) 036/223177 Greetings to all my friends all over the world!!!

★ ★ Switch your Amiga on ★ ★ habe Top Games und suche gute Tauschpartner! Schickt eure Listen an: Wilfried Weitgasser, Vinzentinum/Baereng. 6, 5020 Salzburg

TEST-DRIVE, Crazy Cars, Bad Cat, Goldrunner, The Big Deal, Tasar, Amiganoid, Fire-Power, Detonator, Mission Glevator usw. Bei Peter 04232/2636 ab 20 h nur Österreich!

■ We seek Swapfriends for the Amiga! ■
If you want to swap the hottest software with us, then write: SCS, J. v. Stolberg Str. 38-2h, 1055 RN, Amsterdam, Holland

Hello Amiga-User! I am looking for Swap-Partners all over the world. Please send List or Disk to: Anders Lillrank, Fredsv 4, S-663 00 Skoghall, Sweden. 100% Answer

★ ★ Hey Amiga-Freaks Schweiz, News! ★ ★
Verkaufe und kaufe Software! Alles zu Spottpreisen! Auch MS-DOS Soft. Kontakte B. Achermann, Bruggmatt, 6248 Alberswil

Amiga Freaks !!! I want to swap newest Software! Write to: Morten Petersen, Stejlbjergvej 3,1, 6000 Kolding, Denmark

Kaufe und tausche Software für den Amiga schickt eure Listen an; Bruno Preti, P.O.Box 296, 4005 Basel (Schweiz), verkaufe 3½" Disks für Fr. 2.50 Stück

★ ★ Für Amiga suche ich/habe ich das Neueste. M. Bijnsdorp, Wolgrasstraat 8, 2165 KH Lisserbroek Holland, Antwortet immer bald/bis bald

I've got hottest stuff!!!!!
Contact me: Patrick Michelotti,
Tellenhofweg 3,
CH-6045 Meggen ■ TCA

Suche: Hardware

Suche Amiga 500 o. 1000 mit Monitor. Nehme günstigstes Angebot. L. Roßkothen, Bergschenweg 21, 4133 Neuk.-Vluyn, Tel. ab 14 Uhr: 02845/2265. verk. 64'er u. Data B.

SUCHE Amiga 2000 (mögl. 2 LW od. 1 LW + HD), Monitor (Stereo) u. Zbh. (u.U. auch LA500). SUCHE auch (Farb-)Drucker (NL-10, NEC P6) oder anderen Tel. 0911/306156

Amiga 500 mit Monitor sowie Drucker günstig zu kaufen gesucht. Tel. 02543/295

★ ★ ★ ★ ★ ★ A M I G A 1000 ★ ★ ★ ★ ★ ★
★ ★ Suche dringend ST-EMULATOR für ★ ★
★ den Amiga 1000. Informationen oder ★
★ Bezugsquelle an Michael, T. 0251/524865

DRUCKER — CITIZEN
Suche Drucker LSP 120 D von Citizen
Preis sollte nicht über 180 DM liegen!!
Angebote an 07805/1780 »Achim«

Suche Amiga 500. Nehme günstigstes Angebot. Jürgen Hamacher, Sandstr. 15
5010 Bergheim 3

Suche gebr. Amiga 500 bis 800,— DM oder im Tausch mit 128 + 1571 + Zubehör (1 Jahr) alt. Manfred Hortenbach, Moltkestr. 8, 4500 Osnabrück, Tel. 0541/432836 ab 17.00

Ausland

Suche AMIGA 1000. Tel.: 0041/81/531789 Mario Zahner/Obere Au 205 V/CH-7220 Schiers

Biete an: Hardware

Amiga 1000 — 768-K-RAM-Erweiterung DM 350,—; **Gemini 10X**-Centronics-Drucker DM 400,—; C64-Software + Literatur bei E. Hecht, POB: 1273; 8482 Neustadt; 09602/2515

S/W-Videokamera NEU mit Anschluß für DIGI-VIEW incl. Wechseloptik 1,6/16 mm DM 580,— A. Willbränder, Heckenerstr. 16, D-5469 Windhagen ★ Tel. 02645/4424

Verkaufe: 1M-Amiga-System für nur 1088 DM: Amiga 500 + 512 K-Erweiterung + Echtz.-Uhr + TV-Modulator für Fernseher + Zubehör. ☏: 02641/1452

VERKAUFE AMIGA 2000 KOMPLETT: Monitor, Tastatur, 2 3½"-Laufwerke, PC-Karte mit 1 5¼"-Laufwerk, umsth. VB: 3700,—, Tel.: 089/347496

Drucker NEC P6 zu verkaufen für ca. 1000 DM, ca. 1 Monat alt, noch Garantie
Tel.: 02842/50398 Hyungho Park

★ ★ ★ **AMIGA 1000** (PAL) ★ ★ ★
512 K, Monitor 1081, Literatur und diverse Software wg. Systemwechsel zu verkaufen. DM 2150 (Festpr.) 06122/52469, 17—18 Uhr

Verkaufe: Commodore DF1: DM 150
Original MCC Pascal DM 100
Original Mirror Hacker Package DM 60
Tel.: 069/2545-498 tags, 06051/72359 abends

Verkaufe Amiga 1000 512 K mit Maus + Grundis für **1400 DM VB. Suche A 500** zahle bis 700 DM ohne Monitor, **mit** bis 1100 DM
Angebote ab 16 Uhr Michael 02173/76041

★ ★ ★ Super ★ ★ ★ Verkaufe PAL AMIGA 1000 mit 1,5 MB, Uhr und df1: incl. Handbücher zu DM 2000,—, auch 10 Raten.
Tel. 02208/73082

AMIGA 2000 mit Monitor (Commodore)
3 3½-Zoll-Laufwerken (2 intern, 1 ext.) und Zubehör zu verkaufen.
Edwin Zinth, Tel. 0991/27100 ab 19.30 Uhr

Verkaufe: 3½"-Laufwerk (NEC 1036A, Slimline); abschaltbar, 1,5 Monate Garantie, Top-Zustand, wenig gebraucht, Preis: 325,— DM. Anrufen bei: Armin Keim, Tel.: 07041/2971

Verkaufe wegen Systemwechsel auf MS-DOS neuwertigen Commodore-RGB-Farbmonitor-1081 für DM 600,— (7 Monate alt). Anrufe bitte unter 07641/1503. Do it and get it

Olivetti M 10, 8 KB, zu verkaufen oder Tausch gegen Amiga-Hard- bzw. Software (nur Originale mit Dokumentation!) Tel. 0221/697611 ab 19 Uhr

Verkaufe: Amiga 1000 (512 KByte) + 1081 + Sounddigitizer + Hardware + Bücher: 1800 DM
1 MByte (intern), somit 1,5 MByte: 400 DM
Anrufen unter 0711/385605. Micha verlangen

Amiga 2000, 1 MB RAM, 2 Floppy 880 K, Farbmonitor, Farbdrucker NEC P7, fast neu, VB 4800 DM. Tel. 07627/8341

A 500; 1 MB abschaltbar; Echtzeit; ext. NEC 1036a m. Schalter & Bus; Staubschutzhaube; Zustand neuwert.; 1600,— o. Mon.; 2300,— m. 1081; 089/712264

Verkaufe Starboard 2-MB-Fast-RAM-Erweiterung für Amiga 1000. VB 1350,— DM incl. Multifunktions-Board (Uhr, Parity-Check, RAM-Disk), Metaschk, Tel. 0711/7655517

Brother Interface-Box IF-50 zum Anschluß v. Brother-Schreibmaschine an Amiga 1000 par. o. ser. Bus. Orig.-Verp. mit Kabel. Absol. neuw. 400 DM, J. Ludwig 069/448929

AMIGA ★ ★ ★ ★ ★ ★ ★ ★ ★ AMIGA
TV Modulator 520: Fast nie gebraucht 35 DM: Deluxe Paint Original 80 DM. Chris Tingay, An der Horst 22/3, 3070 Nienburg

AMIGA-1000-Besitzer??? Lust auf PC???
Dann Tel.: 0721/698944! Amiga-Sidecar (512 K + eingebaute Amiga-Clock) für 1300,— DM zu verkaufen

Verkaufe Sidecar 1060 + 512 KB + 8087 + 2 LW 5¼" für 1500,— DM. R. Vogel, Am Hagdorn 41, 7518 Bretten

■■■■ HALLO FREAKS !!! ■■■■
Verk. Akustikkoppler CDI Hitrans 300 = 150 DM u. eine 256-KB-Speichererweiterg. A1000 =140 DM. Michael 0591/64557 ab 19 Uhr

Suche: Amiga 2000 <2000 DM, Software
Monochrom Monitor <200 DM
Markus Preuss/An der Kopfbuche 10/5024
Pulheim-Stommeln/Tel. 02238/13641

Der Wahnsinn!! Amiga 1000 auf 1 MB aufrüsten nur 250 DM. Super Sounddigitizer für Amiga 500/1000/2000 beste Qualität 120 DM mit Soft + Anl. Tel. 0431/711302 18—20 Uhr

Amiga Laufwerk (Amigo) 250,– Sidecar 3 Mon. 900,– ProText (MS-DOS) org. mit Lexikon 200,– Textomat-Amiga org. 50,–, VBM-Text org. 50,– wegen Systemwechsel, Tel. 02841/72645

★ ★ ★ ★ ★ BRANDHEISS ★ ★ ★ ★ ★
AMIGA 500 1 MB + Monitor
Resco-Modem für C-64 SW-Monitor
★ ★ ★ ★ ★ Tel. 07054/2568 ★ ★ ★ ★ ★

Genlock A8600 mit Trafo, für Amiga 500/1000/2000, Neupreis 1750,— DM (+200,— für Umbau) für nur 1250,— DM zu verkaufen. Telefon 0721/553435

■ Verkaufe Commodore Amiga und Monitor
■ Verkaufe Bücher zum Amiga und C64 ■
■ Verkaufe Zubehör zu beiden Computern ■
■ Tel. 08253/6744 von 17 bis 17.30 Uhr ■

Amiga 2000: neue org. Commodore MS-DOS, PC-XT-Karte, macht Ihren 2000er IBM-kompatibel, mit 5¼-Zoll-Laufwerk + Softw. kompl. Tel.: 06033/63387

Verschiedenes

Suche & vermittle Kontakte von Computerbesitzern. Info gegen 80 Pf. Rückporto bei: BERNHARD WILFERT, Albrecht-Dürer-Str. 3, D-8510 Fürth (es lohnt sich).

HILFE: Wer zeigt mir gegen Bezahlung die letzten Tricks mit »De Luxe Music«, »Future Sound« und »Sound Scape Pro Midi Studio« Tel. 08071/7609 Raum Rosenheim

Was steht wo über den Amiga? Umfangreiche Superbase-Datei. Info gegen Freiumschlag. Goedecke, Hafenstr. 19, 3301 Walle

Clubgründung im Raum Ostfriesland!
Meldet Euch!
Anrufe 19 bis 20 Uhr: 04971/1210
Auch Ossis beherrschen die Technik ...!

Suche billiges AMIGA INTERN von DATA BECKER, Tel. 09401/2239 ab 18 h

SUCHE: 1. Musikdemos (z.B. Sonix Dreams 1-3), **2.** Die Bilder des Grafikwettbewerbes, **3.** Kontakt zu Amiga-Usern im Raum **Willich** zwecks Softwaretauschs. 02154/4345 (Andi)

Suche Leute, die auch in Assembler programmieren oder gut malen können, zwecks Erfahrungsaustausch bzw. Zusammenarbeit
M. Wieler Schloßstr. 84, 4053 Jöchen 5

Usergruppe Moers sucht noch weitere Mitglieder mit Inter. an Hard + Software
Info: W. Hoerhold, 02845/33052, ca. 17—21 h

BERLIN: Suche Kontakte zwecks Programm- und Erfahrungsaustausch.
Möglichst in Berlin.
Tel. 030/8032563 (abends)

Neueröffnung der Berliner-Box
Parameter 8/1/N/V Tel. 6041323

■ Suche Suche Suche ■
Zeitschrift AMIGA Ausgabe 8/9 87
Zahle Neupreis!
Tel.: 07322/4228 (ab 18 Uhr)

Amiga-500-Anwender sucht Kontakte im Raume Aachen zwecks (Erfahrungsaus-)Tausch. Tel. 0241/67153 (Gerhard)

Video-Animation! Spitzen-Anwender gesucht für unseren »Club der Besten!« Wir haben Zugang zu Profi-Technik (Schnittstudio, Riesenprojektion)! Köln (0221) 211538

AMIGA 1000 WIE NEU!
Das AMIGA HANDBUCH (Markt & Technik) Neupreis 49 DM für 30 DM zu verkaufen. Richter, 2900 Oldenburg, Tel. 0441/83978

Kaufgemeinschaft! Raum Stuttgart und Umgebung. Gründe Amiga PD-Kaufgemeinschaft, Edgar Tabar, Haldenrainstraße 175, 7000 Stuttgart-Rot

AMIGA Literatur für Programmieren:
Data Becker Maschinensprache 30,— DM
M&T Programmier Handbuch 40,— DM
M. Wieler Schloßstr. 84, 4053 Jüchen 5

★ ★ ★ Ricercar e.V. ★ ★ ★
Der führende Verein für alle AMIGA, ST und C 128-User. Wir bieten mehr! Zeitschrift, PD-Software, Beratung, etc. Info: Ricercar e.V., Dorstener Str. 31, 435 Recklinghausen, Tel.: 02361/15943

Suche Kontakte zu netten AMIGA-Anwendern, auch Neulingen, zwecks Erfahrungs- und Programmtauschs. Ich nutze mein Gerät privat und geschäftlich. Michael Lange, Boppstr. 4, 1000 Berlin 61, Tel. 030/6931028

AMIGA Computer-Markt

Achtung:

Wir machen unsere Inserenten darauf aufmerksam, daß das **Angebot**, der **Verkauf** oder die **Verbreitung** von urheberrechtlich geschützter Software nur für Originalprogramme erlaubt ist.

Das Herstellen, Anbieten, Verkaufen und Verbreiten von **»Raubkopien«** verstößt gegen das Urheberrechtsgesetz und kann straf- und zivilrechtlich verfolgt werden. Bei Verstößen muß mit Anwalts- und Gerichtskosten von über DM 1 000,— gerechnet werden.

Originalprogramme sind am Copyright-Hinweis und am Originalaufkleber des Datenträgers (Diskette oder Kassette) zu erkennen und normalerweise originalverpackt. Mit dem Kauf von Raubkopien erwirbt der Käufer auch kein Nutzungsrecht und geht das Risiko einer jederzeitigen Beschlagnahmung ein.

Wir bitten unsere Leser in deren eigenem Interesse, Raubkopien von Original-Software weder anzubieten, zu verkaufen noch zu verbreiten. Erziehungsberechtigte haften für ihre Kinder.

Der Verlag wird in Zukunft keine Anzeigen mehr veröffentlichen, die darauf schließen lassen, daß Raubkopien angeboten werden.



Schlägel u. Eisen Str. 46 · 4352 Herten · Tel. 0 23 66/5 51 76

SOUND A 500/1000/2000	
Sounddigitizer für A 500/1000/2000	98.-
Sounddigitizer Bausatz	65.-
MIDI A 500/1000/2000	
Midi-Schnittstelle nach DIN	98.-
Midi-Schnittstelle Bausatz	65.-
BOOTSELEKTOR Elektronisch A 500/1000/2000 macht aus Ihrem Laufwerk DF1: ein bootfähiges Laufwerk. Auf Wunsch auch DF2 und DF3	48.-
BOOTSELEKTOR Bausatz incl. Aufbauanleitung	32.-
AMIGOS 3 1/2" Einzel Floppy Metallgehäuse, Amigos Bus, Ein-/Aus-Schalter	369.-
AMIGOS 3 1/2" Doppel Floppy Metallgehäuse, Amigos Bus, Ein-/Aus-Schalter	698.-
AMIGOS 5 1/4" Einzel Floppy Metallgehäuse, Amigos Bus, Ein-/Aus-Schalter 40/80 Track Umschaltung MS-Dos fähig	448.-
AMIGOS KOMBI Doppel Floppy 3 1/2" und 5 1/4" Metallgehäuse, Amigos Bus, Ein-/Aus-Schalter 40/80 Track Umschaltung MS-Dos fähig	795.-
AMIGOS 3 1/2" Einzel Floppy ATARI-Laufwerk, Metallgehäuse, Ein-/Aus-Schalter, Netzteil	389.-
AMIGOS 5 1/4" Einzel Floppy ATARI-Laufwerk, Metallgehäuse, Ein-/Aus-Schalter, Netzteil	468.-

NEU NEU NEU NEU NEU NEU NEU NEU NEU NEU NEU NEU NEU

Für IBM XT/AT oder Kompatible 3 1/2" Laufwerke für das Neue IBM-Format 720 KByte wie es in den Neuen IBM-Rechnern der PS-2 Serie verwendet wird.
Für IBM PS-2 externe 5 1/4" Laufwerke zum Übertragen Ihrer Daten/Programme die noch im alten Format vorliegen.
– PREIS AUF ANFRAGE –. Bitte bei allen Bestellungen Gerätetyp (Amiga 500 etc.) mit angeben.

Ecosoft Economy Software AG

Kaiserstraße 21, D 7890 Waldshut, Tel. 077 51 - 79 20
Villa Domingo, CH 6981 Astano, Tel. 091 - 73 28 13

Prüf-Software und Frei-Programme (fast) gratis

Über 3'000 Disketten mit professioneller Prüf-Software (Programme vor Anwender-Registrierung prüfen), aber auch nützlichen Amateur-Programmen für den beruflichen und privaten Gebrauch für IBM-PC/Kompatible, Macintosh, Atari ST, Amiga, C64/128, Apple II.

Katalog auf Disketten und Verzeichnis DM 10.-
(Bitte Computermodell angeben und Banknote oder Scheck beilegen.)

Gegen Einsendung dieses Inserates erhalten Sie zusätzlich einen Gutschein für 1 Gratis-Ecosoft-Diskette.

Neu: Emulation von Fremdsoftware, z.B. MS-DOS auf Amiga, C64 auf Amiga, Macintosh auf Atari ST, usw. Gratis Info.-Schrift verlangen. 161

Private Kleinanzeigen

Ausland

Amiga-users with a lot of soft (like me) and Special interests for Video (film) and education: write me: W. Bogaerts, Heiststwg, 6, 2880 Putte-Beerzel (Belgium)

Amiga Contacs wanted:
3 Clover court, Murston, Sittingbourne, kent me 10 3 QW, — England — Tel. (004479577553)

Gewerbliche Kleinanzeigen

AMIGA-Finanzbuchhaltungssoftware
Testdiskette/Softwareprobe für 15,— bei MICROTEC, Rigaweg 1, 3300 Braunschweig, Tel. 05309/1466 — Weitere Händleranfragen erwünscht!

PUBLIC DOMAIN BLITZVERSAND
Blitzversand (= 2 Tage)
Über 200 Disks! Disk unter 5.01 DM!!!
Liste gratis bei A. Gauger Software
Buhlstr. 16a, 7505 Ettlingen
Tel. 07243/31828

Digitron-Datentechnik
Ihr neuer Partner rund um den AMIGA
Dauerniedrigpreise bei Soft- u. Hardware
Liste anfordern 0561/824256 o. 05674/4168

RISIKO! Das lustige Ratespiel für bis zu 6 Spieler. Super Grafik, voll Mausgesteuert. Mit Editor! 20,— DM-Schein an: M. Naldini, Postfach 1137, 8210 Prien

Gesucht: selbstgeschr. AMIGA-Programme. Gute Bezahlung! Softwarevertrieb/-verkauf H. Friebe, Herthastr. 26, 8000 München 19

Gewerbliche Kleinanzeigen

Amiga-Soft Public-Domain 300 Disketten. Fred Fish-Disketten bis Nr. 95, Faug, Auge, Amicus, Panorama, Amuse usw. Die Programme werden ausschließlich auf 2.D-Disketten kopiert. Preis pro Diskette ab **4,90 DM**
Leerdisketten No.Name 2.DD ab **24,— DM**
Markendisketten Nashua 2.DD ab **29,— DM**
Laufwerk anschlußfertig 1036A **229,— DM**
Original Golem-Box 2 MB bei mir **898,— DM**
Sonstige Hardware aller Art auf Anfrage!
Gnoth's Copier-Service Dietmar Gnoth
4300 Essen 1 ——— **Tel. 0201/281301** ———

AMIGA PD Soft-Kopie 0,80 DM od. auf 2DD NoName 3,70 DM, Tel. 02327/89293

DIGI-Serv CE, Digitalisierung ihrer Bilder in allen Auflös. (auch HAM), viel Grafik-Softw. (Orig.) u. Neupr., Infos anf. bei Juris-Grafik, Bahnhofstr. 106, 6392 Neu-Anspach

Amiga Software zum kleinen Preis!
Analysieren Sie ihre Mitmenschen über deren Namen. Sie kommen zu einem verblüffenden Ergebnis.
Cagliostro's Zahlenmagie 25 DM
(Vorkasse bar/Scheck)
Software Studio Plieth, 5000 Köln 80
Berg.Gladb.Str. 696 Tel. 0221/6802868

Fish-Disks Nr. 1-30 nur 120 DM incl. Disks, Porto u. Verpackung! V-Scheck an: Rudolf Sondermaier, Parkstr. 38, 8025 Unterhaching

SUPER-Farbfotos von AMIGA-IFF-Grafiken
Erstklassige Color-Fotoabzüge in allen Größen fertigen wir in kürzester Zeit von beliebigen IFF-Grafiken an. Info und Muster anfordern bei A. Willbrander, Heckenerstr. 16, 5469 Windhagen, T.: 02645/4424

PUBLIC DOMAIN für Amiga & IBM
Tiefstpreise + 24-Std.-Versand
Katalogdisk gegen 5,— Vorkasse
Funkcenter Mitte GmbH
Klosterstr. 130, 4000 Düsseldorf 1
Tel. 0211/362522
Mailbox 0211/360104 —— 18 bis 9 Uhr

Niedrigpreise

3.5"-Drive (NEC 1036A) – komplett anschlussfertig – Frontblende & Metallgehäuse amigafarben – abschaltbar	**275.-**
5.25"-Drive – komplett anschlussfertig – 40/80 Tracks – abschaltbar	**350,-**
AMIGA 2000	**2250.-**
mit Monitor 1084	**2890,-**
Monitor 1084 entspiegelt	**725.-**
3.5"-Drive intern	**250.-**
NEC P6	**1190.-**
NEC P6 Color	**1550.-**
Golem Box 2MB	**899.-**

Stalter Computerbedarf
0631/24285
Kurt-Schumacher-Strasse 6
6750 Kaiserslautern

Tips und Tricks

Kommen Sie mit Ihrem Amiga zurecht? Nutzen Sie seine Fähigkeiten optimal? Benötigen Sie Hilfe bei einer schweren Aufgabe? Der richtige Tip zur rechten Zeit spart Ihnen viel Arbeit. Hier finden Sie eine ganze Menge »Zeitspartricks«.

Jeden Monat stehen wir vor der Aufgabe, aus den zahlreichen Einsendungen zu dieser Rubrik die auszuwählen, die schließlich veröffentlicht werden. Dies ist oft nicht leicht. Gerade wenn zu einem Thema mehrere, ähnliche Einsendungen kommen, fällt es schwer, den richtigen herauszupicken. Viele Leser haben sich die Mühe gemacht, nur den Vorschlag eines einzigen können wir drucken. Die anderen mögen uns nicht böse sein. Vielleicht klappt es beim nächstenmal.

Versuchen Sie es mit einem neuen Trick. Sie haben sicher im Laufe der Zeit immer mehr entdeckt, was auch anderen Lesern hilft. Sei es ein Trick zur Programmierung, die bessere Bedienung eines Programms, ein Kniff, um ein bestimmtes Spiel zu lösen, oder eine kleine Bastelei. Nahezu alles ist für uns und die Leser des AMIGA-Magazins von Interesse.

Primzahlenberechnung in Basic schnell, schneller, am schnellsten

Dieses Programm gibt alle Primzahlen bis zu einer wählbaren Grenze aus. Es beruht auf dem »Sieb des Erathostenes«. Dies ist die schnellste Methode, um Primzahlen zu berechnen.

Natürlich ist die Geschwindigkeit auch eine Frage der Programmierung. Das Beispielprogramm benötigt für die Ausgabe aller Primzahlen bis 1000 nur 5 Sekunden:

```
PRINT "Primzahlsieb des Erathostenes"
PRINT "============================": PRINT
INPUT "Obergrenze der Primzahlenberechnung ";pmax
d%=(pmax-1)/2:s%=(sqr(pmax)-1)/2
CLS :PRINT "Alle Primzahlen bis"pmax":"
DIM pz%(d%) :PRINT :PRINT 2; :a%=1 :t1=TIMER
FOR i%=1 TO s% :IF pz%(i%)=0 THEN PRINT 2*i%+1; :a%=
                      a%+1 :IF a% MOD 10=0 THEN PRINT
 FOR j%=2*i%*(i%+1) TO d% STEP 2*i%+1:pz%(j%)=1:
                              NEXT:NEXT: t2=TIMER
FOR i%=s%+1 TO d% :IF pz%(i%)=0 THEN PRINT 2*i%+1;
                :a%=a%+1 :IF a% MOD 10=0 THEN PRINT
NEXT: t3= TIMER
PRINT: PRINT "Anzahl :"a%
PRINT"Rechenzeit"t2-t1"Sekunden"
PRINT"Zeit insgesamt"t3-t1"Sekunden"
```

Um die Rechenzeit herabzusetzen, wurde auf eine Strukturierung des Programms verzichtet. Es benötigt die meiste Zeit zur Ausgabe der Zahlen. Die reine Rechenzeit beträgt für die Ermittlung der Primzahlen bis 1000 weniger als eine Sekunde.

Haben Sie eine schnellere Lösung in Amiga-Basic?

(Frank Pählke/ub)

Diskettenmanipulationen

Mit einem Diskettenmonitor können Sie die Daten auf den Disketten erforschen und direkt verändern. Zum Beispiel können Sie auf einer leeren Diskette Blöcke als belegt kennzeichnen:

Auf jeder formatierten Diskette gibt es einen als »BitMap« bezeichneten Block, der wichtige Informationen über den freien Speicherplatz einer Diskette enthält. Die Blockadresse der »BitMap« steht im »Root Block« (Spur 40, Block 0). Der gesuchte Pointer befindet sich in Spalte 8, Zeile 10 (Diskwik-Monitor). Es handelt sich um das achtzigste Langwort im »Root Block«.

Haben Sie die »BitMap« gefunden?

Sie hat folgenden Aufbau:

— Im ersten Langwort steht die Prüfsumme des gesamten Blocks
— Jedes der folgenden 55 Langwörter repräsentiert den Zustand von je 16 Blöcken auf der Diskette, angefangen mit Nummer 2 bis zum Block 1759.

Zur Erinnerung: Auf jeder Diskette befinden sich 1760 Blöcke mit jeweils 512 Byte. Dies ergibt eine Gesamtkapazität von 1760 x 512 Byte = 880 KByte.

— Die restlichen 72 Langwörter der »BitMap« haben keine Funktion, gehen aber in die Prüfsumme mit ein.

Wollen Sie einen Block als belegt kennzeichnen, so müssen Sie eine »0« an der entsprechenden Stelle eintragen. Einen freien Block erkennt das Betriebssystem durch ein gesetztes Bit »1« im verantwortlichen Langwort.

Jetzt können Sie belegte Blocks als frei kennzeichnen. Dies führt dazu, daß beim Speichern von Daten diese Blöcke eventuell überschrieben werden. Deshalb die Warnung, arbeiten Sie immer mit Sicherheitskopien.

Natürlich können Sie auf der Diskette noch weitere Informationen finden und verändern. Sie können beispielsweise die »Protectionbits« einer Datei direkt auf der Diskette ändern. Also nichts wie ran an die Disketten, um weitere Geheimnisse zu entlüften.

(Cord-Henning Scholz/ub)

Menüprogrammierung

Die Programmierung von Menüs hat einen Haken. Wird während der Basic-Programmentwicklung eine Fehlermeldung ausgegeben, so ist die Basic-Menüleiste nicht mehr sichtbar.

Im Direktmodus schafft der Befehl MENU RESET Abhilfe. Um diese Prozedur in der Testphase eines Programms — in der Fehler erfahrungsgemäß recht häufig auftreten — zu vereinfachen, können Sie den Aussprung aus einem Programm per Basic-Menü mit der gleichzeitigen Freigabe des Menüs verbinden. Die Befehle hierzu lauten:

```
MENU 1,1,1 "Quit"
IF MENU(0)=1 AND MENU(1)=1 THEN MENU RESET:
                          SCREEN CLOSE 1:END
```

Dieser Befehl gibt sowohl das Basic-Fenster als auch die Menüleiste frei.

(Dieter Hofer/ub)

Übersichtliche Kommentare

Mit dem CLI-Befehl FILENOTE kann jeder Anwender eine Notiz zu einer Datei schreiben, die in einer Kommentarzeile steht. Diese Kommentarzeile erscheint nur beim LIST-Befehl im Anschluß an die übrigen Daten. Allerdings erscheint der Kommentar zu einer Datei in einer gesonderten Zeile und verschiebt die gesamte Dateiauflistung:

```
test DIR rwed 30-MAY-87 02:41:29
old DIR rwed 30-MAY-87 02:43:30
dieser Kommentar zu File old stört
nächste DIR rwed 04-JUN-87 12:12:12
```

Mit Hilfe von Steuerzeichen im Text können Sie Kommentare auch direkt hinter der Uhrzeit erscheinen lassen. Geben Sie, um eine Bemerkung zu einer Datei zu schreiben, den Befehl in dieser Form ein:

```
FILENOTE test "{<CTRL,k>,7*<CTRL,i>,3*<SPC>}
Kommentar"
```

Dieser Trick erlaubt es Ihnen, zumindest kurze Kommentare direkt hinter dem Datum einzufügen. Bei Eingabe von weniger als sieben <CTRL i> (= TAB) können Sie auch längere Bemerkungen einfügen, ohne die Zeilenlänge zu überschreiten. Sie überschreiben dann allerdings das Datum beziehungsweise die anderen Anzeigen.

(Bernd Zimmermann/ub)

Grüße aus dem Herzen des Amiga

Das Betriebssystem des Amiga befindet sich im Kickstart-RAM ($FC0000 - $FFFFFF beziehungsweise 16580608 - 16777216). Basic-Programmierer können mit Hilfe des Befehls PEEK einen Blick in die Systemroutinen werfen. Schauen Sie sich in diesem Bereich um. An manchen Stellen ist auch ohne Disassembler etwas erkennbar. Mit dem folgenden Basic-Programm finden Sie eine besondere Nachricht:

```
REM Werte für Kickstart 1.1
Start = 16653596 : Ziel = 16653672
REM Werte für Kickstartversion 1.2
REM Start = 16649670 : Ziel = 16649747
CLS
WIDTH 67
PRINT
FOR i = Start TO Ziel
  PRINT CHR$(i);
NEXT i
```

(Rolfdieter Schiedrum/ub)

Aus Farbe wird Schwarzweiß

Es ist bei Farbbildern häufig problematisch, genau vorherzusagen, wie die Farben in einer Schwarzweiß-Hardcopy auf einem Drucker erscheinen. Das Basic-Programm »Grau« erstellt von einem Farbbild eine Schwarzweiß-Kopie. Dies hilft Ihnen abzuschätzen, wie ein Farbbild auf einem Schwarzweiß-Drucker wiedergegeben wird. Um die korrekten Grauwerte zu berechnen, summiert das Programm von allen 32 Farben die Rot-, Grün- und Blauanteile. Daraus berechnet es den Mittelwert, der den dargestellten Grauton repräsentiert. Sie erhalten eine Grauabstufung, die der Helligkeit der Farben entspricht.:

```
GRAU:
CT=WINDOW(7)        * = Zeiger auf Windowstruktur
CT=CT+46            * Adresse + 46
CT=PEEKL(CT)        * = Zeiger auf Screenstruktur
CT=CT+48            * Adresse + 48
CT=PEEKL(CT)        * = Zeiger auf Colormapstruktur
CT=CT+4             * Adresse + 4
CT=PEEKL(CT)        * = Zeiger auf Farbtabelle
' Sie können die Berechnung des Zeigers
' auf die Farbtabelle auch in einem Befehl
' zusammenfassen:
' CT=PEEKL(PEEKL(PEEKL(WINDOW(7)+46)+48)+4)
' Die Aufteilung erfolgt nur zur besseren
' Erläuterung
FOR N=0 TO 31
  GruenBlau=PEEK(CT+2*N+1)
  Gruen=INT(GruenBlau/16)
  Blau=GruenBlau-Gruen*16
  Rot=PEEK(CT+2*N)
  A=(Gruen+Blau+Rot)/48
  PALETTE N,A,A,A
NEXT N
END
```

Das Listing läßt sich in jedes Basic-Programm als Unterroutine einbauen. Sie müssen in diesem Fall den Befehl RETURN anhängen. »Grau« liefert zwar noch keine identische Darstellung dessen, was ein Drucker aus einem Farbbild macht, aber die Wiedergabe ist besser als beispielsweise die Gründarstellung des Amiga-Monitors.

Wenn Sie mit der Grünschaltung arbeiten, sind einige Punkte und Flächen schwarz, obwohl sie in der Farbdarstellung mit hoher Helligkeit erscheinen. Dies liegt daran, daß in dieser Betriebsart der Monitor tatsächlich nur die Grün-Anteile des RGB-Signals herausfiltert. Jeglicher Rot- beziehungsweise Blauanteil spielt keine Rolle. »Grau« berücksichtigt die Helligkeit aller RGB-Werte. Probieren Sie dies einmal mit einer Farbgrafik aus.

(Peter Weiland/ub)

CLI-Fenster ohne Rahmen

Für das Problem »CLI-Fenster ohne Rahmen« gibt es eine einfachere und kürzere Lösung als das Programm »Border« in der Ausgabe 10/87 auf Seite 92:

Um statt der normalen 77 die vollen 80 Zeichen Breite zu nutzen, läßt sich der Rahmen eines CLI-Fensters einfach mit den ESC-[-Befehlen ausschalten. Escape-Befehle werden normalerweise zur Druckersteuerung verwendet. Beim Amiga werden sie vor der Ausgabe an den in den Preferences eingestellten Drucker angepaßt. Diese Anpassung wird aber auch bei der Ausgabe im CLI-Fenster durchgeführt. Wird beispielsweise von einem Programm in einem CLI-Fenster die Sequenz <ESC [4m> ausgegeben, erscheinen danach alle Texte unterstrichen. Dies ist auch über die Tastatur möglich. Drücken Sie nacheinander die Tasten <ESC>, <[>, <4> und <m>. Jetzt ist Unterstreichen eingeschaltet. Im AMIGA-Magazin 8/9, Seite 86, steht ein Teil der weiteren Befehle. Es gibt aber nicht nur Befehle für die Einstellung der verschiedenen Darstellungsarten, sondern auch für die Formatsteuerung. Hierzu einige Beispiele:

Tastenkombination	Bedeutung von n
ESC [n u	Breite in Zeichen
ESC [n x	Linker Rand in Pixeln
ESC [n y	Abstand zur Oberkante des Fensters
ESC [n t	Anzahl der Zeilen
ESC c	normales Format

Wie Sie aus der Tabelle sehen, kann der gesamte Bildschirm durch wenige Eingaben genutzt werden. Bringen Sie zunächst das CLI-Fenster mit der Maus auf die Größe des Bildschirms. Mit dem Befehl »ESC [80u« wird die Zeilenbreite auf 80 Zeichen gestellt und der linke Rand mit »ESC [0x auf die äußerste linke Position gesetzt. Um in dem großen CLI-Fenster aufzuräumen, kann es mit <CNTL L> gelöscht werden. Sofort verschwindet auch der Rand des Fensters. Schließlich sollten Sie noch <Return> drücken, damit die Formatbefehle nicht als Teil des nächsten CLI-Befehls interpretiert werden.

Ab jetzt wird bei der Textausgabe das gesamte Fenster benutzt. Wünschen Sie wieder das alte Format, genügen die Tasten <ESC c> <Return>. Jeder, der seinen Bildschirm häufig auf diese Art vergrößern will, kann die benötigten Escape-Sequenzen auch in zwei Textdateien »Rahmen-Aus« beziehungsweise »Rahmen-Ein« zusammenfassen. Dann genügt der einfache Aufruf »TYPE Rahmen-Aus«

(Winfried Stappert/ub)

Auch ohne Uhr: So haben Sie immer die richtige Zeit

Ein Tip für jeden, der keine eingebaute Uhr besitzt und dennoch immer das genaue Datum benötigt. Dieser Tip erspart Ihnen nicht, das Datum beim Starten des Amiga einzugeben, aber Sie brauchen die Arbeit nicht nach jedem Reset zu wiederholen. Ergänzen Sie die »Startup-Sequence« um die folgenden Befehle:

```
FAILAT 30
ECHO "derzeitiges Datum"
DATE
DATE ?
DATE
DATE to datefile
```

Zunächst gibt der Amiga das gespeicherte Datum aus. Das zweite DATE zeigt die Optionen des Befehls und wartet auf die Eingabe eines Datums. Sie können nun das aktuelle Datum eingeben. Stimmt das angegebene Datum bereits, drücken Sie <Return>. Ihre Eingabe wird vom dritten DATE gelesen und mit dem letzten Befehl auf der Diskette gespeichert. Damit ist »datefile« die zuletzt gespeicherte Datei auf der Diskette. Bei jedem Start wird deren Datum automatisch als aktuelles Datum verwendet. So brauchen Sie immer nur einmal pro Tag das Datum eintippen, auch wenn Sie den Amiga zwischenzeitlich ausschalten.

Ist das Datum aktualisiert, können Sie am folgenden Tag die Korrektur des Datums mit der Eingabe »tomorrow« abkürzen.

(Rainer Klier/ub)

Erweiterung zum Vokabeltrainer

Der Vokabeltrainer aus Ausgabe 8/9 läßt sich problemlos erweitern. Eine kleine Änderung des Basic-Programms erlaubt es Ihnen, kurze Erläuterungen hinter den Vokabeln einzufügen, die nicht abgefragt werden. Bei einer richtigen Antwort listet das Programm die zusätzlichen Kommentare mit auf.

Jetzt können Sie Erklärungen, beispielsweise in welchem Zusammenhang eine Vokabel verwendet wird, in der Vokabeldatei eingeben. Genauso wie es in einem guten Wörterbuch Brauch ist. Um Ihr Programm zu ändern, sollten Sie das Listing aus der Ausgabe 8/9 zur Verfügung haben. Folgende Änderungen müssen Sie anhand der Zeilennummern durchführen:

— Zeile 121 ergänzen durch: PRINT
— zwischen Zeile 121 und 122

```
FOR ii= 1 to 3 : Color 1,0
  PRINT "" Bedeutung$(i,ii)
NEXT
```

— In Zeile 122
»CALL Pause (1)« ändern in »CALL Pause (2)«

(Wenn einer Vokabel eine Erläuterung folgen soll, muß hinter der Vokabel ein Leerzeichen und die offene Klammer folgen. Anstelle der Klammer ist ein anderes Zeichen verwendbar. Sie müssen dieses Zeichen im Listing entsprechend ersetzen).

```
- nach 97 IF Eingabe$=CHR$(27) THEN RETURN
   vorhanden   = INSTR(Bedeutung$(index1,1),"(")
   IF vorhanden=0 THEN skip1
   Bedeutung1$ = LEFT$(Bedeutung$(index1,1),
             vorhanden-2)
  skip1:
   vorhanden   = INSTR(Bedeutung$(index1,2),"(")
   IF vorhanden=0 THEN skip2
   Bedeutung2$ = LEFT$(Bedeutung$(index1,2),
             vorhanden-2)
   skip2:
   vorhanden   = INSTR(Bedeutung$(index1,3),"(")
   IF vorhanden=0 THEN skip3
   Bedeutung3$ = LEFT$(Bedeutung$(index1,3),
             vorhanden-2)
  skip3:
nach 98 COLOR 2,0 : LOCATE 7,3
99 IF (Eingabe$ <> Bedeutung$(index1,1) AND
      Eingabe$ <> Bedeutung$(index1,2) AND
      Eingabe$ <> Bedeutung$(index1,3) AND
      Eingabe$ <> Bedeutung1$ AND
      Eingabe$ <> Bedeutung2$ AND
      Eingabe$ <> Bedeutung3$ OR
      Eingabe$ =  CHR$(139)  THEN
nach 342 FOR ii = 1 TO 3
   vorhanden = INSTR(Bedeutung$(i,ii),"("):IF
                 vorhanden=0    THEN skip 4
   Bedeutung1$(i,ii)=LEFT$(Bedeutung$(i,ii),
                              vorhanden-2)
skip4:
343 IF Eingabe$=Bedeutung1$(i,ii) OR Eingabe$=
    Bedeutung$(i,ii) THEN COLOR 2,0:PRINT:
    PRINT Vokabel$(i);: COLOR 1,0:
    PRINT" - ";:COLOR3,0:
    PRINT Bedeutung$(i,1);
- nach Titel:
SCREEN 1,640,200,3,2
DIM Vokabel$(maxindex),Bedeutung$(maxindex,3),
Vokabelstatus(maxindex), Bedeutung1$(maxindex,3)
```

Haben Sie alle Änderungen durchgeführt?

Jetzt können Sie alle Vokabeln mit beliebigen und sinnvollen Kommentaren ergänzen, wodurch das Training noch effektiver wird. Aber nur bei einer richtigen Antwort listet das Programm Ihre Kommentare auf.

(Gerd Pickard/ub)



CLI-Befehle in der RAM-Disk

Wer mit dem CLI arbeitet, wird früher oder später einen der zahlreichen Tips aufgreifen, die Befehle des CLI in die RAM-Disk zu kopieren. Diesem Thema widmen sich zahlreiche Einsendungen. Hier nun die neueste Version:

Die RAM-Disk wird meist in der »Startup-Sequence« automatisch angelegt, die CLI-Befehle kopiert und mittels ASSIGN beziehungsweise PATH zugewiesen. Eine Neuerung, die das Arbeiten komfortabler macht, ist folgende:

Zerlegen Sie das C-Directory in zwei Teile:

1. Die wichtigen Befehle, die Sie laufend brauchen. Kopieren Sie die Befehle in ein Directory »c/c2«.
2. Weniger häufig gebrauchte Befehle. Diese Befehle verbleiben im Directory »c«.

Beim Booten kopiert der Amiga dann nur die wichtigen Befehle in die RAM-Disk. Der Vorteil: Kein unnötiger Speicherplatzbedarf, keine Unzahl von Copy-Befehlen in der »Startup-Sequence« und alle häufig verwendeten Befehle sind immer parat.

Damit der Amiga die Befehle im Directory »c2« auch ohne aktivierte RAM-Disk erkennt, fügen Sie in der »Startup-Sequence« als erste Zeile diesen Befehl an:

```
PATH c/c2 ADD
```

beziehungsweise, wenn Sie den Befehl »PATH« im Directory c2 abgelegt haben:

```
c2/PATH c/c2 ADD
```

Die Initialisierung der RAM-Disk erfolgt in der zweiten Zeile der »Startup-Sequence«:

```
EXECUTE RAMCLI-ONOFF
```

Dieser Befehl funktioniert nur, wenn Sie das nötige Batch-File mit Namen »RAMCLI-ONOFF« mit dem Editor erzeugt haben:

```
IF EXISTS ram:c2
  PATH df0:c/c2 RESET
  DELETE ram:c2 all quiet
ELSE
  MAKEDIR ram:c2
  COPY sys:c/c2 to ram:c2 all
  PATH ram:c2 RESET
  ENDIF
```

Diese Befehlsdatei hat den Vorteil, daß Sie mit einem weiteren Aufruf über EXECUTE die RAM-Disk jederzeit löschen können.

(C.Becker/ub)

Super-Hochauflösung

Überall steht geschrieben, der Amiga hätte eine maximale Bildschirmauflösung von 640 x 512 Punkten. In Wahrheit steht Ihnen eine weit höhere Auflösung zur Verfügung. Ein Screen kann bis zu 704 x 564 Punkte groß sein. Die 69376 zusätzlichen Punkte können Sie mit Hilfe der Grafikbefehle des Betriebssystems ohne weiteres nutzen. Besonders in C und Assembler ist dies recht einfach. Zwei Zahlen müssen geändert werden:

Um einen Screen zu öffnen, müssen Sie eine »NewScreen«-Struktur erstellen. Der dritte Wert dieser Tabelle gibt die Screenbreite an, der vierte die Höhe. Üblicherweise stehen dort, beispielsweise für einen hochauflösenden Screen im Interlace-Modus die Werte 640 und 512. Ersetzen Sie diese Werte durch 704 beziehungsweise 564.

```
NewScreen.Width=704;
NewScreen.Height=564;
```

Wenn Sie nun die Funktion »OpenScreen« anspringen, erscheint der übergroße Screen. Er ist aber nur zum Teil sichtbar. Um dem abzuhelfen, müssen Sie folgendes ändern:

Erstens sollten Sie das Bild des Monitors etwas verkleinern und neu zentrieren. Die Regler hierzu befinden sich im allgemeinen an der Rückseite des Monitors.

Zweitens müssen Sie das Bild mit Hilfe der »Preferences« nach oben links verschieben. Klicken Sie hierzu das große Rechteck in der Mitte des »Preference-Windows« an und bewegen das gesamte Fenster mit der Maus in die gewünschte Ecke. Für die genaue Einstellung ist ein wenig Experimentieren notwendig. Sie dürfen das Bild nicht zu weit verschieben, da sonst der obere Rand des Screens verschwindet. Wenn Sie den Monitor einmal eingestellt haben, können Sie in Ihren Programmen sofort die »Super-Hochauflösung« ausprobieren.

Der Effekt läßt sich im übrigen auch in allen anderen View-Modi ausnutzen, um mehr Punkte darzustellen. So stehen Ihnen zum Beispiel für einen LowRes-Screen 352 statt der üblichen 320 Punkte in der Horizontalen zur Verfügung.

Warum nicht gleich die volle Auflösung?

Die Einschränkung auf 640 x 512 Punkte erfolgt, weil nicht alle Monitore diese extreme Auflösung darstellen können (Overscanning). Besonders an den Rändern kommt es zu Unregelmäßigkeiten. Außerdem sind in den Grenzbereichen nicht alle Sprites sichtbar.

(Jürgen Brendel/ub)

Bildschirmsplitting

Diese kleine Farbdemonstration splittet den Bildschirm in zwei Hälften. Das Programm demonstriert die Abfrage der Position des Rasterstrahls mit Hilfe der Hardware-Register.

```
 execbase = 4
 forbid = -132
 permid = -138
start:
 movem.l d0-d2/a1/a2/a6,-(SP)
 lea $dff004,a1        ; u.a. Y-POS des Rasterstrahls
 lea $dff180,a2        ; Farbregister
 move.l execbase,a6
 jsr forbid(a6)        ; Multitasking aus
zeilenloop:
 move.l #$2b00,d0      ; >>2b<< = Zeile für Split
                       ; >>1b<< = Minimum
Farbloop:
 move.w #$0000,d1      ;erste Farbe
wait:
 move.l (a1),d2        ;
 and.l #$0001ff00,d2; y-Pos. in d2
 cmp.l d0,d2           ; Zeile erreicht ?
 bne wait
 move.w d1,(a2)        ; neue Farbe ins Register
wait1:
 move.l (a1),d2
 and.l #$0001ff00,d2
 cmp.l d0,d2
 beq wait1             ; nächste Zeile abwarten
 cmp.b #$bc,$bfe001
 beq done              ;linke Maustaste gedrückt ?
 add.w #$100,d1        ;neue Farbe
 cmp.w #$1000,d1       ;Farbe wechselt laufend
 blt wait
 add.l #$0100,d0       ;Splitzeile +1
 cmp.l #$13900,d0      ;größer Maximum?
 blt farbloop
 bge zeilenloop
done:
 move.l execbase , a6
 jsr permid(a6)
 movem.l (sp)+,d0-d2/a1/a2/a6
 rts
```

Das Programm ist mit dem Seka-Assembler geschrieben. Nach dem Start wird zunächst jeder weitere Task gesperrt. Danach verändert das Programm ab einer bestimmten Bildschirmzeile die Hintergrundfarbe. Zur besseren Demonstration wird diese Farbe laufend geändert. Nach Durchlaufen der Farbskala erhöht das Programm die Zeile, in der der Bildschirm gesplittet werden soll. Sie können den Programmablauf jederzeit durch einen Druck der linken Maustaste stoppen. Ein Nachteil des Programms ist, daß der Prozessor auf eine bestimmte Rasterposition wartet und nichts anderes erledigen kann. Noch besser wäre es daher, die Farbe des Hintergrunds mit Hilfe des Coppers ab einer bestimmten Position zu verändern, aber das ist eine andere Geschichte.

(Klaus Kuphal/ub)

MS-DOS-Emulator startet durch

Auch MS-DOS verfügt über eine »Startup-Sequence«, die Sie verändern können. Wie? Das zeigt dieser Tip.

Nach dem Starten des MS-DOS-Emulators mit Kickstart 1.1, Transformer 1.0 und MS-DOS-Systemdiskette 2.11, arbeitet der Amiga 1000 mit dem amerikanischen Tastaturtreiber. Geben Sie »AMIGATAST« ein, so erhalten Sie die deutsche Tastaturbelegung. Aus dem Prompt »a:/« wird »a:ö«. Dies läßt sich vermeiden. Bei jedem Start der MS-DOS-Systemdiskette wird eine kleine Batch-Datei mit Namen »Autoexec.bat« durchlaufen — genau wie die »Startup-Sequence« des Amiga. Mit dem Befehl »Type Autoexec.bat« können Sie sich den Inhalt dieser Datei ansehen. In der ersten Zeile steht die Anweisung »ECHO off«. Fehlt diese Zeile, zeigt der Computer alle Befehle beim Booten auch auf dem Bildschirm an. In der folgenden Zeile steht der Befehl »Prompt pg«. Diese Anweisung ist für den Prompt verantwortlich. Um den Befehl zu ändern, müssen Sie den Editor aufrufen. Dieser ist im Gegensatz zum »Amiga-ED« nicht ganz so komfortabel, aber er reicht:

```
EDLIN AUTOEXEC.BAT
```

Als Prompt erscheint » * «. Durch Eingabe der Zeilennummer 2 wird die gewünschte Zeile angezeigt. Um die Änderung vorzunehmen, müssen Sie hinter »2*« die neue Zeile eingeben. Tippen Sie hier ein Leerzeichen ein, so wird als Prompt nur noch »A>« benutzt. Um nach dem Start direkt über die deutsche Tastaturbelegung zu verfügen, empfiehlt es sich, eine weitere Zeile in der Batchdatei einzufügen. Mittels Eingabe von »1i« (1 Zeile, insert) werden Sie nach der eingefügten Zeile mit der Nummer 3 gefragt. In dieser Zeile können Sie »AMIGATAST« eingeben. Analog können Sie weitere Zeilen einfügen, um bestimmte Programme aufzurufen oder Meldungen in der Startsequenz auszugeben. Sie verlassen den Editor mit »END«. Da diese Datei nur beim Start durchlaufen wird, drücken Sie gleichzeitig die Tasten <CTRL> und <ALT,links> und <.> (auf dem Zehnerblock) und starten Ihre MS-DOS-Systemdiskette neu. (Heinz Lösch/ub)

Scrollen im Listfenster

Das Listfenster des Amiga-Basic läßt sich auf mehrere Arten scrollen:

— Am bekanntesten ist das Scrollen mit Hilfe der Cursortasten.
— Seitenweise läßt sich das Fenster durch gleichzeitiges Drücken der Shift-Taste und der Cursortasten nach oben beziehungsweise unten verschieben.
— Die dritte Methode ist hingegen nicht so bekannt. Das Listfenster können Sie auch mit der Maus bewegen.

Um eine bestimmte Zeile auf dem Bildschirm anzuzeigen, brauchen Sie nur die linke Maustaste drücken und bei gedrückter Taste mit der Maus nach unten oder oben aus dem Fenster zu fahren. Das Listing bewegt sich so lange in die geforderte Richtung, bis Sie den Mauszeiger wieder ins Fenster bringen, oder die Maustaste loslassen.

Diese Funktion dient eigentlich zum Markieren eines Blocks, der nachfolgend mit »COPY, CUT und PASTE« bearbeitet werden soll. Um den selektierten Block normal darzustellen, genügt ein Mausklick. Alles ist wie gehabt, nur der Cursor ist genau dort, wo Sie es möchten. (Heinz Lösch/ub)

Auch die Maus läßt sich in Basic kontrollieren

Eine kurzes Listing demonstriert Ihnen, wie Sie die Maus von Basic aus programmieren können. Das Programm zeigt die horizontale und vertikale Position der Maus auf dem Bildschirm an, sobald Sie die linke Maustaste betätigen:

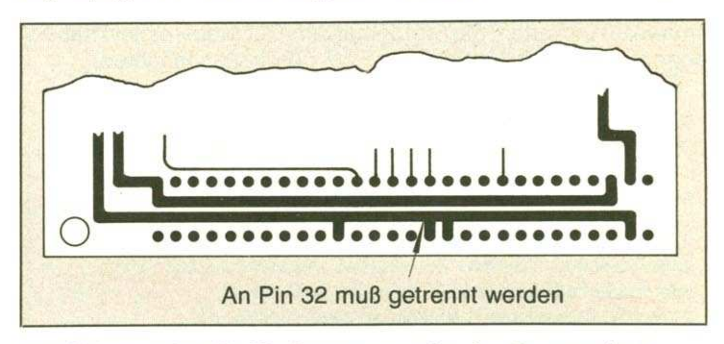
```
WHILE 1
  x=MOUSE(2)
  y=MOUSE(3)
  IF MOUSE(0) > 0 THEN PRINT x;y
WEND
```

Diese Schleife können Sie durch Drücken der Tasten <CTRL> und <C> verlassen. (Eike Cornelius/ub)

RAM-Erweiterung abschalten

Einige Programme für den Amiga 500 laufen nicht mit der Speichererweiterung. Auch das Programm »NoFastMem« hilft in manchen Fällen nicht. Sie müssen die RAM-Erweiterung schon komplett abschalten. Hierzu ist lediglich der Einbau eines Schalters erforderlich:

— Nehmen Sie zunächst die Erweiterungsplatine und betrachten sie von der Unterseite. Trennen Sie auf der Platine die eingezeichnete Leiterbahn direkt an Pin 32 (Bild) durch und überbrücken die Unterbrechung mit einem Schalter. Vor jedem Aus- und Einschalten der RAM-Erweiterung schalten Sie auf jeden Fall den Amiga aus. Warten Sie mit dem Umschalten etwa 10 Sekunden, bis alle Signalpegel im Rechner abgebaut werden.



Die Skizze zeigt die Platinenunterseite der Commodore-Speichererweiterung für den Amiga 500

Die Umbauanleitung bezieht sich auf die Original-Speichererweiterung von Commodore, das Prinzip ist aber bei allen Speichererweiterungen dasselbe: Sie müssen die Verbindung zwischen den Leitungen EXRAM Pin(32) und GROUND (unter anderem Pin 33) durchtrennen und mit einem Schalter überbrücken. Nur wenn EXRAM auf low liegt, erkennt der Amiga die RAM-Erweiterung. Dies ist der Normalfall bei eingebauter Speichererweiterung. Ist die Verbindung bei eingebauter Erweiterung jedoch unterbrochen, steht der zusätzliche Speicher nicht zur Verfügung.

(Anmerkung der Redaktion: Bei ausgeschalteter Erweiterung ist die Verbindung zwischen EXRAM und GROUND unterbrochen. Nur das zusätzliche RAM wird beim Starten nicht aktiviert. Eine auf der Erweiterung eingebaute Uhr arbeitet normal.

Denken Sie bitte daran, daß Sie durch Arbeiten an der Hardware jegliche Garantieansprüche verlieren und Ihren wertvollen Computer zerstören könnten. Eine Umbauanleitung, bei der Sie sich das Löten auf der Platine ersparen, finden Sie in der 68000er Ausgabe 11/87 auf Seite 101). (Frank Ahlhorn/ub)

Tolle Bilder ohne Aufwand

Es lassen sich auch ohne großen Aufwand herrliche Bilder auf dem Amiga erzeugen. Ein Beispiel liefert ein Basic-Programm. Der künstlerische Wert ist auf jeden Fall nicht zu übersehen:

```
DEF FN a(x)=y-SGN(x)*SIN(ABS(b*x-c))
kx=300:ky=170:COLOR 2,0
PRINT "Eingabe der Parameter!"
INPUT "Anzahl der Punkte";az
INPUT "Parameter a,b,c";a,b,c:CLS
SCREEN 1,640,400,2,4
PALETTE 1,0,0,0
WINDOW 2,"Grafik",(1,1)-(670,370),15,1
WINDOW OUTPUT 2:PRINT a,b,c
FOR t= 1 to az
  PSET (x+kx,y+ky)
  xx= FN a(x):y=a-x : x=xx
NEXT t
PRINT "Fertig" :END.
```

Geben Sie das kurze Basic-Programm ein und probieren Sie unterschiedliche Werte für die Parameter aus.

(Nils Oestmann/ub)

CLI-Mate ohne ».fastdir«

Mit Hilfe eines Basic-Programms läßt sich eine der schlechten Angewohnheiten von »CLI-Mate« vermeiden. »CLI-Mate« ist ein leistungsfähiges Tool, mit dem Sie Ihre Disketten verwalten können. Das Programm erlaubt es, Files aus jedem Dateiverzeichnis in andere zu kopieren. Des weiteren erzeugt das Programm für jedes Dateiverzeichnis auf einer Diskette ein »Fast-Directory«. Dieses wird automatisch auf jede Diskette kopiert, deren Schreibschutz nicht aktiviert ist.

Der große Vorteil dieser Prozedur: Jedesmal wenn Sie mit »CLI-Mate« ein Directory von der Diskette lesen, das bereits als ».fastdir« vorhanden ist, zeigt der Amiga prompt das Inhaltsverzeichnis an.

Der Nachteil: Diese zusätzlichen Files nehmen Platz weg. Sie stören auf Disketten, die Sie einmal mit »CLI-Mate« erstellt haben, aber nie wieder mit dem Programm bearbeiten möchten.

Das Programm »Modify.CLI-Mate« schafft Abhilfe:

```
DEFINT a-z
OPEN"r",#1,"df0:c/climate",1

REM  Befindet sich das Programm in einem anderen
REM  Pfad, so muß dieser eingesetzt werden.
FIELD #1,1 AS d$
h$="nil:"+STRING$(4,0)
REM h$=".fastdir" * für alten Zustand*
Loop:
 READ S: IF s=-1 then CLOSE : END
 FOR i = s to s+7
   LSET d$=MID$(H$,i-s+1,1) : PUT#1,I
 NEXT I
GOTO Loop

DATA &H083B
DATA &H0F5E
DATA &H0F8A
DATA &H1780
DATA &H17A9
DATA &H438D
DATA &H4552
DATA &H4BD3
DATA &H4C91
DATA &H4DCB
DATA &H -1
```

Arbeiten Sie immer nur mit einer Sicherheitskopie von »CLI-Mate«. Diese können Sie mit »Modify.CLI-Mate« ändern. Legen Sie die Programmdiskette im Laufwerk df0: ein und starten Sie das Basic-Programm. Es verändert nun alle ».fastdir«-Strings und ersetzt diese durch einen String, der ins »Nichts« führt (NIL:).

Nach der Änderung werden keine ».fastdir«-Dateien mehr angelegt. Alle schon bestehenden werden von »CLI-Mate« angezeigt und können gelöscht werden.

Jetzt werden natürlich alle Directories im Normaltempo geladen. Um ein modifiziertes Programm wieder in den alten Zustand zu versetzen, müssen Sie die im Listing angegebene Zeile von »Modify.CLI-Mate« ersetzen.

(H.Rückeshäuser/ub)

Editieren mit Komfort

Wenn Sie mit dem CLI arbeiten und den Editor mit ED aufrufen, können Sie zwar editieren, aber sonst keine weitere Arbeit mit dem CLI ausführen. Wer beispielsweise während des Editierens einmal den Namen einer Datei benötigt, steht vor dem Problem, daß das CLI auf keine Kommandos reagiert. Abhilfe schafft der folgende Befehl:

```
RUN ED name
```

Nun arbeitet der Editor als eigenständiger Task. Im CLI-Fenster kann normal weitergearbeitet werden. Durch diesen Trick sparen Sie sich das Speichern und erneute Laden der zu editierenden Datei, falls Sie kurzfristig mit dem CLI arbeiten wollen.

(Norbert Cohen/ub)

Read Error, halb so wild

Kennen Sie die Systemmeldung »Read Error Can't Validate Disk«?

Sie kommen dann nicht mehr an die Daten auf der Diskette. Aber nicht verzagen, es gibt eventuell eine Rettung:

Kopieren Sie das gesamte Dateiverzeichnis »l« von einer Workbench in die RAM-Disk:

```
MAKEDIR ram:l
COPY df0:l ram:l all
ASSIGN L:  ram:l
```

Mit Hilfe der Zuweisung durch den ASSIGN-Befehl sucht der Amiga den Disk-Validator nicht mehr auf der eingelegten Diskette, sondern in der RAM-Disk. Legen Sie nun Ihre defekte Diskette erneut ein. Sollte der Amiga diesmal die Diskette ohne Fehlermeldung annehmen, kopieren Sie schnell alle benötigten Dateien auf eine andere Diskette. (Henning Müller/ub)

Amerikanische Tastatur, alles o.k.

Besitzer des Amiga 1000 kennen das Problem: Ein Programm benutzt den deutschen Zeichensatz und Sie haben nur eine amerikanische Tastatur. Wo liegt der Doppelpunkt, wo das Semikolon?

Entweder Sie lernen die deutsche Tastaurbelegung auswendig oder benutzen Tastaturaufkleber. Die dritte Möglichkeit ist mit wesentlich weniger Aufwand verbunden:

Drücken Sie das gewünschte Zeichen auf der amerikanischen Tastatur zusammen mit der <ALT>-Taste. (Marco Trauner/ub)

Reverse in Basic

Um in Basic revers zu drucken, müssen Sie mit dem Befehl COLOR die Hintergrundfarbe ändern. Beispiel:

```
COLOR 1,3 : PRINT "hallo" : COLOR 1,0
```

Der zweite COLOR-Befehl wählt die ursprüngliche Hintergrundfarbe. (Eike Cornelius/ub)

Spiele von der Workbench laden

Viele Spiele, zum Beispiel »Defender of the Crown« oder »Shanghai«, verwenden eigene Zeichensätze. Wenn Sie diese Spiele von der Workbench starten wollen, muß der entsprechende Zeichensatz (Font) vorher in das Dateiverzeichnis »Fonts« auf der Startup-Diskette kopiert werden. Die Arbeit lohnt sich. Nun können Sie jederzeit die genannten Spiele von der Workbench starten, ohne Ihren Amiga mit den Originaldisketten booten zu müssen. (Edwin C. Wirth/ub)

Neuer Befehl für Abfrage im CLI

Das neue Kommando ASK erlaubt es Ihnen, bei der Ausführung einer Befehlsdatei (Batchfile) mittels EXECUTE eine Eingabe von der Tastatur abzuwarten. Bisher gab es hierfür keine direkte Funktion, jedoch einige raffinierte Tricks. Auf den Workbench-Versionen ab 33.560 ist eine solche Funktion bereits installiert. Hinter dem Befehl ASK steht ein Text. Wird der Befehl ausgeführt, gibt die Funktion den Text aus und wartet auf eine Tastatureingabe des Benutzers. Abhängig davon, ob mit <y> (für yes = ja) oder einer beliebigen anderen Taste geantwortet wird, setzt der Amiga einen Fehlercode.

Eine positive Antwort kann mit dem Befehl »IF WARN« abgefragt werden. (ub)

Joystickabfrage in Basic

Die Basicfunktion »STRIG(n)« liefert je nach Argument (n) den Status des Feuerknopfes eines wählbaren Joysticks. Die Funktion übergibt eine »-1« oder »0« als Rückgabewert, je nachdem, ob die getestete Bedingung erfüllt ist oder nicht. In der Beschreibung im Basic-Handbuch hat sich der Guru eingeschlichen. Dort stehen die Rückgabewerte »1« beziehungsweise »0«.

(Jochen Kümmel/ub)

Professionelle Zahleneingabe

Wenn mit der Maus Zahlenwerte in einem bestimmten Bereich eingestellt werden sollen, benutzt man sogenannte Schieberegler (Proportional-Gadgets).

Leider sind diese Regler nicht einfach zu programmieren, aber nach unserem Beispielprogramm werden Sie keine Probleme mehr damit haben.

Ein Beispiel für Schieberegler kennen Sie sicher auch, nämlich die Regler zum Einstellen der verschiedenen Farbanteile bei Zeichenprogrammen. Dabei werden drei Regler verwendet, die einen Bereich von 0 bis 15 abdecken. Natürlich kann dieser Wertebereich auch ganz anders aussehen.

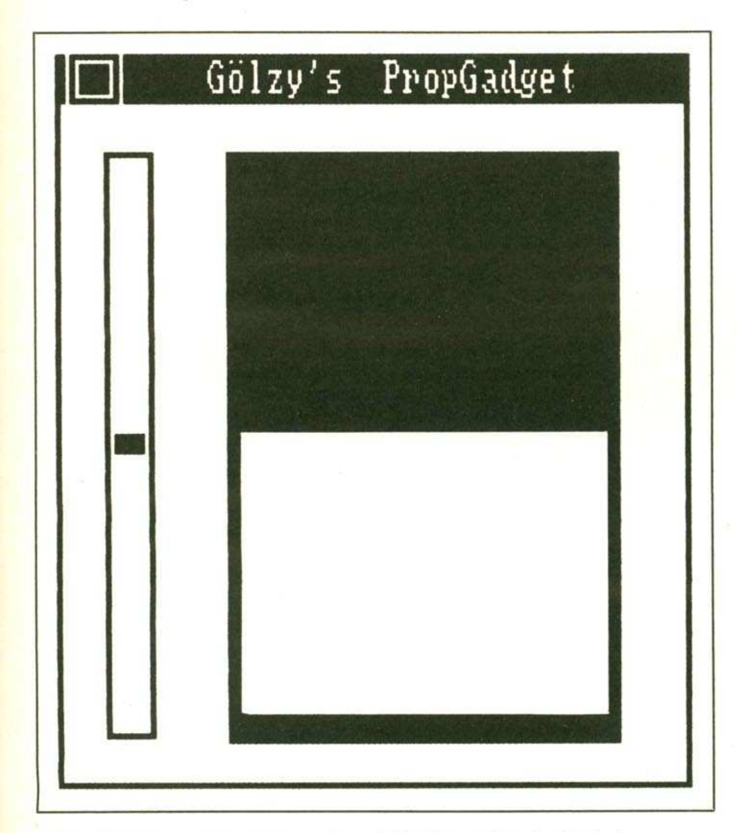


Bild. So sieht das Proportional-Gadget des Beispielprogramms aus. Änderungen sind leicht möglich.

Am besten tippen Sie zuerst das Programm (siehe Listing) ab und testen es etwas. Der rechte Farbblock dient nur zur Demonstration, daß das Programm die Werte auch richtig erfaßt. Man könnte auch die entsprechende Zahl ausgeben lassen.

Zum besseren Verständnis wird das Programm nun ausführlich Zeile für Zeile erläutert.

Zeile 6:	Das Headerfile intuition.h, welches die notwendigen Strukturen und Vereinbarungen enthält, wird eingebunden.
Zeile 7:	Das Ergebnis der Division HEX FFFF(DEZ 65536) durch n ergibt die Anzahl von n Schieberstellungen (hier 100) eines Proportional-Gadgets.
Zeile 8 bis 12:	Zeiger auf notwendige Strukturen werden vereinbart.
Zeile 13:	Die Strukturvariable border wird definiert.
Zeile 14 bis 19:	Initialisierung der PropInfo-Struktur, sie enthält Informationen über unser Proportional-Gadget.
Zeile 15:	Die Flags eines Proportional-Gadgets. Hier wurde ein Vertikal-Gadget mit vordefiniertem Knopf (Schieber) vereinbart.
Zeile 16:	Die Komponenten HorizPot und VertPot. Wir verwenden nur VertPot. Die Startposition des Knopfes wird festgelegt. Es soll zu Beginn in der Mitte des Gadgets stehen, wir halbieren also einfach die Anzahl der Positionen.
Zeile 17:	HorizBody und VertBody. Auch hier wird nur VertBody berücksichtigt. Die Anzahl der Schieberstellungen wird festgelegt.
Zeile 18:	Diese Werte werden von Intuition selbst gesetzt.
Zeile 20 bis 30:	Die Strukturvariable prop_gad der Gadget-Struktur wird initialisiert.
Zeile 21:	Zeiger auf das nächste Gadget im gleichen Fenster. Es wurde kein weiteres Gadget definiert.
Zeile 22:	Die Position der linken und oberen Ecke sowie die Breite und Höhe des Gadgets werden festgelegt.
Zeile 23:	Flags werden keine benötigt. Durch Setzen der beiden »Activations« teilt uns Intuition mit, wenn das Gadget gewählt wird.
Zeile 24:	Der Typ des Gadgets wird festgelegt — hier ein Proportional-Gadget.
Zeile 25:	Ein Zeiger auf eine Borderstrukur. Intuition übernimmt das Zeichnen der Umrandung eines Proportional-Gadgets selbst. Dies kann jedoch durch Setzen des Flags PROPBORDERLESS in der PropInfo-Struktur verhindert werden.
Zeile 26:	Wir verwenden kein Alternativ-Image, keinen Gadget-Text und keine Bedingungen unter Gadgets.
Zeile 27:	Ein Zeiger auf die PropInfo-Struktur.
Zeile 28:	Die Gadget-Kennummer. Diese kann vom Programmierer festgelegt werden.
Zeile 29:	Wenn der Programmierer noch spezielle Daten in Verbindung mit diesem Gadget gespeichert hat, kann hier ein Zeiger auf diese Daten stehen.
Zeile 31 bis 44:	Die Initialisierung der NewWindow-Strukturvariablen FensterVereinbarung.
Zeile 32:	Die Position der linken und oberen Ecke sowie die Breite und Höhe des Fensters werden festgelegt.
Zeile 33:	Die Farben für Detail- und Blockpen werden in dieser Programmzeile vereinbart.
Zeile 34:	IDCMP-Flags. Wir wünschen Informationen über die Anwahl des Close-Gadgets und des Proportional-Gadgets.
Zeile 35:	Die gesetzten Flags des Fenster besagen, daß es beim Öffnen gleich aktiv ist und daß ein Schließsymbol (Close-Gadget) installiert ist.
Zeile 36:	Ein Zeiger auf das erste Gadget im Fenster — hier ein Zeiger auf unser Proportional-Gadget.
Zeile 37:	Ein Zeiger auf die Image-Struktur eines Häkchens für Menüs, wenn einem das von Intuition zur Verfügung gestellte nicht schön genug ist.
Zeile 38:	Der Text, der in der Titelleiste des Fensters erscheinen soll.
Zeile 39:	Ein Zeiger auf eine eigene Screen-Struktur. Wir verwenden den bereits existierenden Workbenchscreen.
Zeile 40:	Ein Zeiger auf eine selbst angelegte BitMap-Struktur.
Zeile 41:	Wenn das Fenster in der Größe veränderbar ist, muß man hier die Mindestwerte eintragen.
Zeile 42:	Der Screen-Typ. Wir benutzen den Workbenchscreen.
Zeile 49:	Die Definition der Funktion openw().
Zeile 51 und 52:	Es wird versucht, die intuition.library zu öffnen. Wenn dies nicht gelingt, wird das Programm in der Funktion closew() beendet.
Zeile 53 und 54:	Hier wird die graphics.library geöffnet, sonst erfolgt der Aufruf der Funktion closew().
Zeile 55 und 56:	Hier wird das Window geöffnet oder wieder closew() aufgerufen.
Zeile 57:	RP erhält die Adresse der RastPort-Struktur unseres neu geöffneten Windows namens fenster.
Zeile 58:	Das Proportional-Gadget wird angelegt. Die Parameter der Funktion Refreshgadgets() sind: — ein Zeiger auf die Gadget-Struktur — ein Zeiger auf das Fenster, in dem sich das Gadget befindet — ein Zeiger auf den Requester, in dem sich das

	Gadget befindet
Zeile 60:	Funktion closew()
Zeile 62:	Wenn die intuition.library bereits geöffnet wurde, wird sie geschlossen.
Zeile 63:	Wenn die graphics.library bereits geöffnet wurde, wird sie geschlossen.
Zeile 64:	Wenn das Window bereits geöffnet wurde, wird es geschlossen.
Zeile 65:	Programmende mit der Funktion exit().
Zeile 67:	Funktion block()
Zeile 68:	Die Definition des formalen Parameters, der die Höhe des Rechtecks angibt.
Zeile 70:	Der Zeichenmodus JAM1 wird festgelegt.
Zeile 71:	Die Farbe des Zeichenstiftes ist weiß.
Zeile 72:	Ein Rechteck wird gezeichnet. Die Parameter von RectFill(): — Die RastPort-Struktur des Fensters, in dem gezeichnet werden soll — die vier Eckpunkte des Rechtecks
Zeile 73:	Die Stiftfarbe wird geändert. Die neue Farbe: orange.
Zeile 74:	Der Block wird in Abhängigkeit des formalen Parameters gezeichnet (siehe auch Zeile 68).
Zeile 76:	Vereinbarung der Funktion message.
Zeile 78:	Die Integervariable class wird die Art der Nachricht aufnehmen.
Zeile 79:	Eine Nachricht wird mittels der Funktion GetMsg gelesen.
Zeile 80:	In class wird die Art der Nachricht gerettet.
Zeile 81:	Die Nachricht wird bestätigt. Intuition löscht diese Nachricht und kann die nächste zur Verfügung stellen.
Zeile 83:	Wenn eine Nachricht empfangen wurde, enthält der Return-Wert die Art der Nachricht, ansonsten null.
Zeile 85:	Die Funktion checkgad()
Zeile 87:	Die Gadget-Struktur found wird vereinbart. Diese wird aufgerufen, wenn ein Gadget gewählt wurde.
Zeile 88:	Die Definition der Variablen foundID, welche später die Kennummer des gewählten Gadgets enthalten wird.
Zeile 89:	found erhält die Adresse der Struktur des gewählten Gadgets.
Zeile 90:	Aus der in Zeile 89 gelesenen Struktur wird die GadgetID gelesen und in foundID gespeichert.
Zeile 91:	Eine switch-Anweisung. Hier wird je nach dem Wert von foundID reagiert.
Zeile 92:	In diesem Fall (case) beinhaltet foundID den Wert 0. Das Rechteck wird gezeichnet (Zeile 93).
Zeile 96:	Keine Reaktion, falls eine andere GadgetID ermittelt wurde.
Zeile 101:	Die Hauptfunktion main().
Zeile 103:	Die Variable class, diesmal für die Funktion main() wird definiert.
Zeile 104:	Die Funktion openw() wird aufgerufen. Libraries und das Fenster werden geöffnet, die RastPort-Struktur wird gelesen und das Gadget gezeichnet.
Zeile 105:	Da der Schieber des Gadgets zu Beginn in der Mitte steht, wird hier der Block ebenfalls in Mittelstellung gebracht (100 Möglichkeiten = > 100/2 = 50).
Zeile 106:	Eine Endlosschleife, die nur unterbrochen wird, wenn das Close-Gadget links oben im Fenster angeklickt wird.
Zeile 107:	Die Funktion message() wird aufgerufen. Eine Intuition-Nachricht wird gelesen. Die Variable class erhält den Return-Wert dieser Funktion.
Zeile 108:	Eine switch-Anweisung. Je nach Art der Nachricht wird reagiert.
Zeile 109 und 110:	Falls ein Gadget gewählt wurde, wird die Funktion checkgad() aufgerufen (Zeile 111).
Zeile 114:	Falls das Close-Gadget, links oben im Fenster angeklickt wurde, wird das Programm mittels der Funktion closew() (Zeile 115) beendet.
Zeile 118:	Keine Reaktion auf andersartige Nachrichten.
Zeile 121:	Ende der switch-Anweisung.
Zeile 122:	Ende der while-Schleife.
Zeile 123:	Ende der Funktion main().

Zum Übersetzen des Programms verwenden Sie bitte für den Aztec-C-Compiler folgende Aufrufe:

```
cc propgad +l -s
ln propgad.o -lm32 -lc32
```

Für den Lattice-C-Compiler geben Sie folgendes ein:

```
lc propgad
blink lib:c.o,propgad.o lib lib:lcm.lib,lib:lc.lib,
lib:amiga.lib
```

Die beim Compilieren auftretenden Warnungen können ignoriert werden.

Nachdem Sie das Programm durchgearbeitet haben, sollte es Ihnen in Zukunft keine Schwierigkeiten mehr bereiten, Proportional-Gadgets selbst zu programmieren. Durch ihren Einsatz erhält das Programm einen professionellen Anstrich. Außerdem können weniger Fehler bei der Eingabe der Werte geschehen.

(Arno Gölzer/rb)

Programmname:	Propgad
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.4 oder Lattice-C V3.1
Aufrufe:	siehe Text

```
Programm : Propgad
------------------

 1 z00 /*
 2 GY8         PropGad
 3 WU1  Demo eines Proportional-Gadget
 4 PM8         Arno Gölzer
 5 8X0 */
 6 Uj   #include <intuition/intuition.h>
 7 EA   #define SCHRITTE (0xFFFF/100)
 8 I2   struct IntuitionBase    *IntuitionBase;
 9 9T   struct GfxBase          *GfxBase;
10 Nc   struct IntuiMessage     *nachricht;
11 Hs   struct RastPort         *RP;
12 F7   struct Window           *fenster;
13 2f   struct Border   border;
14 AU   struct PropInfo prop={
15 Ba8          FREEVERT| AUTOKNOB,
16 MM           0, 0xFFFF/2,
17 Ed           0, SCHRITTE,
18 q6           0,0,0,0,0,0
19 cY0 };
20 iH   struct Gadget prop_gad={
21 Rt8          0,
22 W6           20,20,20,120,
23 JU           0,GADGIMMEDIATE| RELVERIFY,
24 1X           PROPGADGET,
25 KO           (APTR)&border,
26 SW           0,0,0,
27 as           (APTR)&prop,
28 Y0           0,
29 zF           0
30 nj0 };
31 1I   struct NewWindow FensterVereinbarung={
32 728          0,0,260,150,
33 hc           0,1,
34 cY           CLOSEWINDOW| GADGETDOWN| GADGETUP,
35 H9           ACTIVATE| WINDOWCLOSE,
36 Mi           &prop_gad,
37 CR           NULL,
38 Gh           "    Gölzy's  PropGadget",
39 ET           NULL,
40 FU           NULL,
41 Fl           0,0,0,0,
42 dw           WBENCHSCREEN
43 Ow0 };
44 g5   /*
45 Aa8          ____________
46 cR9           Funktionen
47 Cc8          ____________
48 pE0 */
49 4f   openw()
50 kD   {
51 VP8          if(!(IntuitionBase = (struct IntuitionBase *)
52 3m0                          OpenLibrary("intuition.library",0)))
                                  closew();
53 TH8          if(!(GfxBase = (struct GfxBase *)
54 TKO                          OpenLibrary("graphics.library",0)))
                                 closew();
55 ZV8          if(!(fenster=(struct Window *)
```

```
56 OjO                              OpenWindow(&FensterVereinbarung)))
                                    closew();
57 ZM8          RP=fenster->RPort;
58 3v           RefreshGadgets(&prop_gad,fenster,NULL);
59 xSO }
60 3C  closew()
61 vO  {
62 UJ8          if(IntuitionBase) CloseLibrary(IntuitionBase);
63 op           if(GfxBase)       CloseLibrary(GfxBase);
64 9h           if(fenster)       CloseWindow(fenster);
65 OW           exit();
66 4ZO }
67 nJ  block(y)
68 8c8          int y;
69 3WO {
70 9k8          SetDrMd(RP, JAM1);
71 sg           SetAPen(RP, 1);
72 X2           RectFill(RP, 70, 20, 230, 140);
73 2s           SetAPen(RP, 3);
74 6M           RectFill(RP, 80, y+30, 220, 130);
75 DiO }
76 Gd  message()
77 Be  {
78 Zo8          int class=0;
79 xf           if(nachricht=(struct IntuiMessage *)GetMsg(fenster->
                UserPort)){
80 k1G                  class=nachricht->Class;
81 TN                   ReplyMsg(nachricht);
82 Kp8          }
83 dG           return(class);
84 MrO }
85 Tu  checkgad()
86 Kn  {
87 9u8          struct Gadget *found;
88 MO           int foundID;
89 tB           found=(struct Gadget *) nachricht->IAddress;
90 MF           foundID=found->GadgetID;
91 Jx           switch(foundID){
92 M5G                  case 0:{
93 e70                          block(prop.VertPot/SCHRITTE);
94 Ra                           break;
95 X2G                  }
96 by                   default:{
97 UdO                          break;
98 a5G                  }
99 b68          }
100 c70 }
101 Kg  main()
102 a3  {
103 xd8         int class;
104 hJ          openw();
105 uq          block(50);
106 W3          while(1){
107 h8G                 class=message();
108 fJ                  switch(class){
109 4WO                         case GADGETUP  :
110 U2                          case GADGETDOWN:{
111 lde                                 checkgad();
112 js                                  break;
113 pKO                         }
114 8n                          case CLOSEWINDOW:{
115 7he                                 closew();
116 nw                                  break;
117 tOO                         }
118 xK                          default:{
119 qze                                 break;
120 wRO                         }
121 xSG                 }
122 yT8         }
123 zUO }
(C) 1987 M&T
```

Listing. »Propgad«, ein Programm zur Demonstration von Schiebereglern (Proportional-Gadgets). Bitte mit dem Checksummer von Seite 66 eingeben.

Strings schnell gefunden

Das Amiga-Basic ist zum Suchen von Zeichenketten oder Zahlen in großen Feldern zu langsam. Was liegt näher, als eine Assemblerroutine zu verwenden, um wesentlich höhere Geschwindigkeiten beim Suchen von Strings zu erreichen?

Bei jeder Art von Datenverwaltung tritt das Problem auf, daß schnell bestimmte Zeichenketten gefunden werden müssen. Entweder man hält das zu durchsuchende Feld ständig sortiert, was viel Zeit kostet, oder besitzt eine schnelle Suchroutine. Unser Assemblerprogramm ist bis zu 50mal schneller als das entsprechende Basic-Programm.

Mit Amiga-Basic ist es praktisch unmöglich, ein solches Programm zu schreiben. Ein Ausweg aus dieser Misere ist die Assemblerprogrammierung. Wie Sie im Listing sehen, ist der Assemblerteil in Form von DATA-Zeilen im Basic-Programm untergebracht. Durch das Unterprogramm »Array.Comp.Einlesen« werden diese Werte in den Speicher geholt.

Um das Assemblerprogramm zu nutzen, muß man sich an folgenden Aufruf halten:

```
CALL Array.Comp.Aufruf&(FeNr&,LStr%,StrV&,StrF&,
Diff%)
```

Die einzelnen Variablen bei diesem Aufruf werden nun im einzelnen vorgestellt.

»Array.Comp.Aufruf&« ist die Anfangsadresse des Assemblerprogramms und wird für die Funktion CALL benötigt.

Die Variable »FeNr&« ist ein Zeiger auf die Adresse der Variablen »FeldNr%«. Er wird benötigt, damit das Assemblerprogramm das Ergebnis dort ablegen kann. Das Basic-Programm kann somit auf diese Variable zugreifen. Findet das Programm keine passende Zeichenkette, steht in der Variablen FeldNr% der Wert der Variablen AbNr% minus 1. Wenn Sie zum Beispiel ab dem Feldelement mit der Nummer 200 suchen lassen und das Ergebnis 199 ist, gibt es diesen String nicht. Das heißt, ab dem String mit dem Index 200 existiert er nicht.

Mit »LStr%« wird die Länge der zu suchenden Zeichenkette übergeben. Durch diese Angabe ist die Maschinenroutine in der Lage, viele Strings von vornherein auszuschließen, da die Länge genau gleich sein muß.

Die Position des zu suchenden Textes befindet sich in der Variablen »StrV&«. Diesen Wert erhält man, indem die Funktion »SADD« auf die Stringvariable angewendet wird. Dies sehen Sie in Zeile 21 des Listings.

Um auch bestimmen zu können, ab welchem Feldelement gesucht wird, übergibt man die Variable »StrF&«. Sie erhält man durch die Funktion »VARPTR«.

Zu guter Letzt wird mit der Variablen »Diff%« die Differenz zwischen dem ersten zu testenden Feldelement und dem höchsten Feldindex festgelegt. Die Suche startet bei dem Element mit dem Index »AbNr%«. Dies macht es möglich, auch nachdem ein passender String gefunden wurde, den Rest des Feldes zu prüfen. Das ist dann nötig, wenn Sie zum Beispiel alle »Mayer« in Ihrer Datei finden wollen. Das Programm sucht bis entweder ein passender String gefunden oder das Ende des Feldes erreicht ist.

Wie man die einzelnen Übergabeparameter bestimmt, sehen Sie im Listing in den Zeilen 19 bis 25.

Um die Routine in eigenen Programmen verwenden zu können, benötigen Sie die Zeilen 44 bis 63 und 19 bis 25. Die Variablennamen können Sie natürlich nach Belieben verändern.

Array-Comp beschleunigt die Suche nach Strings enorm. Sie können also in Zukunft viel Zeit sparen, wenn Sie String-Felder durchsuchen müssen, und das Arbeiten mit Ihren Programmen wird zum Vergnügen. (Georg Brünsing/rb)

Programmname:	Arraycomp
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2

```
 1 AyO CLEAR,100000&:DEFINT a-z                    Programm : Arraycomp
 2 15  INPUT "ab ";AbNr%                           --------------------
 3 5f  DimF%=5000
 4 WJ  FeldNr%=0
 5 WQ  Diff%=0:Zeit!=0:Array.Comp.Aufruf&=0:StrV&=0:Lstr%=0
 6 Dd  FeNr&=0:StrF&=0
 7 UZ  StringVariable$="ABC 123 def"
 8 rq  SuchVariable$="ABC 123 def"
 9 Ih  GOSUB Array.Comp.Einlesen
10 m6  a=5:n=15
11 v2  PRINT "Es wird ein Stringarray von der Größe DIM(";DimF%;") an
       gelegt"
12 on  PRINT "und mit Stringlängen zwischen";a;"und";n;"belegt."
13 HB  DIM StringFeld$(DimF%)
14 ho  FOR i=0 TO DimF%-1
15 re2   StringFeld$(i)=SPACE$(INT(RND*(n-a+1))+a)
16 LQ0 NEXT
17 3R  PRINT "Der Wert der gesuchten Variable wird jetzt in das"DimF%
       " Feld"
18 U1  PRINT "gelegt und"
19 1b  StringFeld$(DimF%)=StringVariable$
20 eE  StringFeld$(2500)=StringVariable$
21 B6  StrV& = SADD(SuchVariable$)
22 aY  Lstr% = LEN(SuchVariable$)
23 1y  FeNr& = VARPTR(FeldNr%)
24 Qw  StrF& = VARPTR(StringFeld$(AbNr%))
25 3J  Diff% = DimF%-AbNr%
26 91  'wenn jetzt noch neue Variablen definiert
27 sH  'werden verschieben sich die Arrays
28 ws  PRINT "Array$-Comp. wird aufgerufen"
29 Bo  Zeit!=TIMER
30 80  Array.Comp.Aufruf&=VARPTR(Array.Comp%(0))
31 731  CALL Array.Comp.Aufruf&(FeNr&,Lstr%,StrV&,StrF&,Diff%)
32 sIO PRINT "gebrauchte Zeit =";TIMER-Zeit!;"Sek."
33 33  PRINT "gefunden in Feld";FeldNr%+AbNr%
34 Jt  PRINT:PRINT
35 Vx  PRINT "und jetzt das gleiche nocheinmal in einer FOR-NEXT Schl
       eife"
36 Iv  Zeit!=TIMER
37 1W  FOR i&=AbNr% TO DimF%
38 kE2   IF StringFeld$(i&)=SuchVariable$ THEN
39 zP4     PRINT "gebrauchte Zeit =";TIMER-Zeit!;"Sek."
40 G5      PRINT "gefunden in Feld";i&:END
41 Z12   END  IF
42 1q0 NEXT
43 RM  END
44 BJ  Array.Comp.Einlesen:
45 az1  DIM SHARED Array.Comp%(50)
46 YY   i&=0
47 pm2   FOR n%=0 TO 50
48 IJ4     READ x%
49 iv0 i&=i&+x%
50 Y74     Array.Comp%(n%)=x%
51 f52   NEXT n%
52 dV0 SumZahl&=463967&
53 WP  IF i&><SumZahl& THEN PRINT "Fehler in Data's":BEEP:LIST Da
       tas: END
54 Eq  RETURN
55 Jp  Datas:
56 U8  ' ***** Maschinen - Code ****
57 mq  DATA  18663, 32760, 17029, 17030, 17031, 9263, 68, 8815
58 AF  DATA  64, 8303, 60, 8751, 56, 9839, 52, 5657
59 Vh  DATA -7861, 5657, 14012,-1,-19901, 26144, 19009, 26414
60 ge  DATA  17028, 6169,-7860, 6169,-7796, 6169, 9284, 14849
61 D6  DATA  10312,-19188, 26120, 21317, 26388, 24822, 22153,-16830
62 Aa  DATA  26382, 5657,-7861, 5657,-8580, 1, 24778, 13959
63 pg  DATA  19679, 8190, 20085
(C) 1987 M&T
```

Listing. »Array-Comp« zum schnellen Finden von Stringvariablen in Feldern mit Demoprogrammteil. Bitte mit dem Checksummer (siehe Seite 66) eingeben.

Der Pseudo-Guru

Heute lösen wir die Aufgabe, die in der letzten Ausgabe gestellt wurde. Aber keine Angst, ein neues »verwanztes« Programm bekommen Sie auch. Und diesmal finden Sie die Lösung sicher schon schneller als das letzte Mal, oder?

Beim letzten Mal hatten wir das Problem, daß unsere Routine leider nicht die zwei erwünschten Sterne gezeichnet hat. Der Fehler steckt dabei, wie so oft, im Detail. In der Unterroutine wurde die Variable »winkel« verändert. Das hat zur Folge, daß auch im Hauptprogramm der Variablen »i« der neue Wert zugewiesen wird.

Diesen Mißstand kann man jedoch leicht beheben. Es gibt nämlich zwei verschiedene Methoden, Parameter an ein Unterprogramm zu übergeben. Zum ersten ist da die Übergabe durch Referenz. Dabei wird ein Zeiger auf die Variable (in unserem Fall i) an das Unterprogramm übergeben. Der Zeiger auf die Variable »winkel« wird dann auf diesen Wert gesetzt und zeigt somit auf denselben Speicherplatz. Dadurch kann die Variable auch in der Unterroutine verändert werden.

Die zweite Methode ist die Übergabe durch den Wert der Variablen. Hierbei wird der Wert der Variablen in die entsprechende Veränderliche der Unterroutine übertragen. Um zu kennzeichnen, daß ein Wert und nicht ein Zeiger übergeben werden soll, setzt man im Aufruf den betreffenden Parameter in Klammern. Bei unserem Programm lautet also der Aufruf korrekt:

```
drawstern xm,ym,(i),ra
```

So, nun behält unser Zähler »i« seinen ursprünglichen Wert und die Schleife wird richtig abgearbeitet. Die Informationen zu den verschiedenen Übergabemethoden finden Sie im Kapitel 6.1 des Basic-Handbuchs noch ausführlicher.

Mittels der Übergabe durch den Wert werden Programme leichter lesbar, da die Variablen des Hauptprogramms unverändert bleiben. Auch vermeidet man damit die sogenannten Seiteneffekte, die oft zu unerklärlichen Fehlern beim Programmlauf führen. Die sichere Methode zur Änderung von Veränderlichen des Hauptprogramms durch ein Unterprogramm ist die »SHARED«-Anweisung, die auch im Kapitel 6 des Basic-Handbuchs erklärt wird.

Sicher haben Sie den Fehler auch selbst entdeckt, aber das nächste Problem liegt schon vor Ihnen. Dieses Mal scheinen beim Programmieren sogar mehrere kleine Fehler passiert zu sein. Doch zunächst soll erwähnt werden, was das kleine Programm überhaupt tun soll.

Das Programm soll eine beliebige Zeichenkette von der Tastatur einlesen, wobei auch Kommas enthalten sein dürfen. Dann soll dieser Text in einer Datei gespeichert werden. Nach dem Laden des Textes sollte dieser dann ausgegeben werden.

Leider funktioniert das Programm nicht so, wie wir das wünschen, aber Sie werden die Fehler sicher recht schnell finden.

```
INPUT"Dateiname : ";datei$
inputstring x$
putstring datei$
getstring datei$
END
SUB putstring a$ STATIC
 OPEN a$ FOR OUTPUT AS #1
 inputstring x$
 PRINT#1,x$
 CLOSE 1
END SUB
SUB getstring a$ STATIC
 OPEN a$ FOR INPUT AS #1
 INPUT#1,z$
 PRINT z$
 CLOSE 1
END SUB
SUB inputstring x$ STATIC
 PRINT "Bitte Text eingeben::"
 FOR i=0 TO 50
  WHILE (c$<>CHR$(13))
   WHILE(c$="")
    c$=INKEY$
    PRINT c$;
   WEND
   x$=x$+c$
  WEND
 NEXT i
END SUB
```

Wenn Sie die offensichtlicheren Fehler entfernt haben, probieren Sie doch einmal den Text »Betrag : 300,00 DM« zu speichern und wieder zu laden. Denn auch dabei versucht der Pseudo-Guru Sie zu überlisten. Aber das schafft er ja sicher nicht! (rb)

Professionelle Abfrage

Professionelle Programme verwenden den sogenannten »File Requester« um einen Dateinamen auszuwählen. Mit »Disk.h« sind auch Sie in der Lage, so ein praktisches Fenster zu benutzen. Sie erhöhen damit die Sicherheit ganz erheblich und außerdem macht Ihr Programm einen besseren Eindruck.

Für C-Programmierer ist »Disk.h« (siehe Listing 1) ein wertvolles Hilfsmittel. Das Directory einer Diskette wird dargestellt, man kann von DF0: auf DF1: wechseln und umgekehrt. Auch das Auf- und Absteigen in Unterverzeichnissen ist ohne Probleme möglich.

Das Directory wird in einem großen Block (siehe Bild) dargestellt. Rechts daneben befindet sich ein Rollbalken. Mit diesem kann man die Einträge nach oben und unten scrollen. Klicken Sie hierzu auf den roten Knopf und »ziehen« Sie ihn nach oben oder unten. Wenn Sie über oder unter den Knopf klicken, werden die Einträge nur um eine Position gescrollt.

Die schwarzen Einträge sind Unterverzeichnisse. Mit einem Klick auf einen Verzeichnisnamen können Sie in das jeweilige Verzeichnis hinabsteigen. Zurück in das übergeordnete Verzeichnis, gelangen Sie mit einem Klick auf das Gadget mit der Aufschrift »CD /«.

Wenn Sie einen Filenamen, welche weiß aufgelistet werden, anklicken, erscheint dieser in dem Textgadget, das sich unter dem großen Block befindet.

Die beiden Gadgets »DF0:« und »DF1:« benötigen Sie, um zwischen den Laufwerken zu wechseln. Ein Klick auf das jeweilige Gadget genügt.

Die Routine wird beendet, wenn Sie »STOP« anwählen. Wenn jedoch »OK« angeklickt wird, gibt die Routine den ausgesuchten Dateinamen, komplett mit Suchpfad, zurück.

Es sind nur zwei Programmzeilen notwendig, um den File Requester auf den Bildschirm zu bekommen:

```
#include "disk.h"
filename=disk();
```

Die erste Zeile bindet die Routine in Ihr Programm ein und mit der zweiten rufen Sie sie auf. Das Beispielprogramm Demo (siehe Listing 2) verdeutlicht dies.

Beachten Sie, daß neben der Funktion disk() auch noch weitere interessante Routinen zur Verfügung stehen. Zum Beispiel print() oder block(). Diese können Sie natürlich auch für andere Anwendungen benutzen. Zu diesem Zweck noch eine kurze Beschreibung der im Programm verwendeten Funktionen.

Die Funktion »openw« (Zeile 46 bis 53) öffnet die Grafik- und die Intuitionbibliothek und macht ein eigenes Fenster auf dem Bildschirm auf. Dabei wird die »NewWindow«-Struktur verwendet, die in den Zeilen davor mit den entsprechenden Werten belegt wird. Die Funktion, um das Fenster und die Bibliotheken wieder zu schließen, ist »closew« in Zeile 54 bis 68. Außerdem liefert sie, wenn der Übergabeparameter ungleich Null ist, den Namen des gewählten Verzeichnisses beziehungsweise der Datei an das aufrufende Programm.

Um Nachrichten von Intuition zu erhalten, dient die Funktion »message« (Zeile 69 bis 78). Sie liefert einen Rückgabewert, der an anderer Stelle ausgewertet wird.

Speicherplatz für Stringvariablen kann mit »strmem« (Zeile 79 bis 87) belegt werden. Der Rückgabewert wird von anderen Programmteilen weiterverwendet.

Das Wechseln des aktuellen Verzeichnisses geschieht mit »cd« in den Zeilen 88 bis 111. Hier ist gut zu sehen, wie eine gute Routine aussehen soll. Für den Fall, daß nicht in das gewünschte Directory gewechselt werden kann, erfolgt die Sicherung des alten Pfadnamens in der Variablen »store«. Somit kann das Programm auch im schlimmsten Fall weiter abgearbeitet werden.

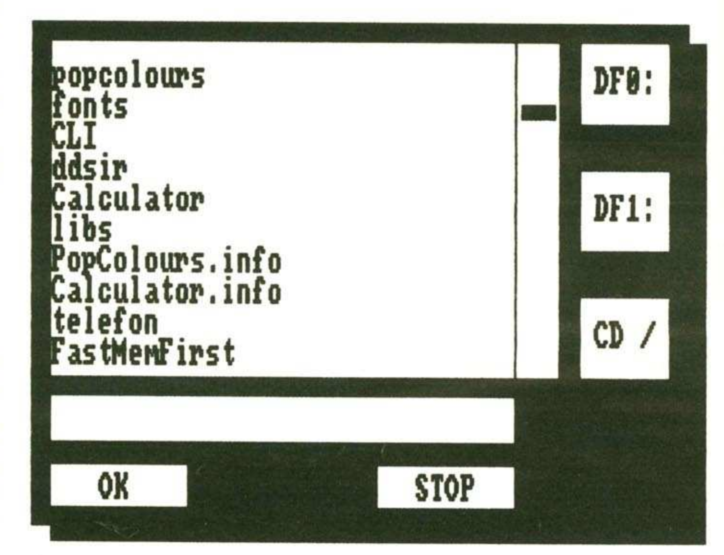
Die Funktion »readdir« (Zeile 112 bis 146) liest, wie der Name schon sagt, ein Directory ein. Durch die »if«-Anweisung in Zeile 129 unterscheidet das Programm, ob es sich um ein Directory oder eine Datei handelt.

Für Textausgabe eignet sich sehr gut »print« (Zeile 147 bis 156), da Sie außer der Position auch die Farbe mit angeben können. Durch zweimaliges Aufrufen mit anderer Farbe und X- und Y-Koordinaten, die jeweils um eins größer sind, erhalten Sie eine sehr schöne Darstellung des Textes.

Auch die Funktion »block« in Zeile 157 bis 164 läßt sich oft einsetzen. Sie zeichnet ein farbiges Rechteck an beliebiger Position mit frei wählbarer Größe und Farbe. Wie bei der Textausgabe erhalten Sie durch einen zweiten Aufruf mit veränderten Parametern einen Block mit Schatten.

Von Zeile 172 bis 196 finden Sie das Unterprogramm »setgad«, das die Gadgets zeichnet und die Texte hineinschreibt. Direkt danach folgt der Programmteil (»checkgad«) zur Auswertung, ob eines der Gadgets betätigt wurde.

Wenn Fehler auftreten, bringt die Funktion »fehler« eine Meldung auf den Bildschirm.



So sieht das Fenster aus, in dem Sie die Dateinamen und Dateiverzeichnisse wählen können

Zum Schreiben der Directory- und Dateinamen in die zwei Anzeigeblöcke findet »Wdir« (Zeile 253 bis 268) Verwendung.

Die letzte Funktion ist »disk«, die die Steuerung übernimmt. Von hier aus werden alle anderen Funktionen aufgerufen. Der Rückgabewert wird, wie bereits weiter oben erwähnt, von der Funktion closew verwendet.

Das Programm Disk.h müssen Sie nicht compilieren, da es mittels »include«-Anweisung in Ihr Programm eingebunden wird. Aber wie immer sollten Sie Ihr Programm mit der »+l«-Option übersetzen, um 32 Bit lange Integer-Variablen zu verwenden.

Disk.h ist eine kurze und sehr nützliche Routine, um sicher und bedienerfreundlich Dateien auszuwählen — und außerdem ist sie nicht die langsamste... (Arno Gölzer/rb)

Programmname:	Demodisk
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.40
Aufrufe:	cc demodisk +l -s ln demodisk -lc32

```
Programm :Disk.h
-----------------

 1 d00 #include <graphics/gfxmacros.h>
 2 uZ  #include <exec/exec.h>
 3 Rg  #include <intuition/intuition.h>
 4 xv  #include <libraries/dosextens.h>
 5 yn  #define MAUSKLICK while((class=message()) != MOUSEBUTTONS);
 6 DC  static int zaehler, fnr, class;
 7 5a  static int farbe[85];   /* Dir oder Filename */
 8 wZ  static int first=0;   /* flag */
 9 CX  static int nr;       /* die Nummer, ab der gelistet wird */
10 Uk  static unsigned char *Ename[85];   /* Name eines Eintrags */

11 2k  static unsigned char filename[32]=NULL;   /* Filename im Gad
       get */
12 PU  static unsigned char undo[32];       /* UNDO Filename im Gadg
       et */
13 Ku  static unsigned char aktuellDir[200];   /* enthält die aktue
       lle Dir */
14 7y  static unsigned char store[200];   /* Sicherheitskopie von a
       ktuellDir */
15 DZ  static unsigned char Rwert[232];        /* Returnwert (Pfad+
       Filename) */
16 XP  static char *bef []={         /* Text der Kommando-'Gadgets'
        */
17 z13    "DF0:",
18 58     "DF1:",
19 u9     "CD /"
20 dZ0 };
21 ku  struct GfxBase        *GfxBase;
22 pI  struct IntuitionBase *IntuitionBase;
23 Fy  static struct IntuiMessage  *nachricht;
24 16  static struct Window        *fenster;
25 A8  static struct RastPort      *RP1;
26 Qm  static struct FileLock      *dir;
27 EB  static struct FileInfoBlock *eintrag;
28 os  static struct StringInfo infoN={    /* Info für Textgadget */
29 wX3    filename,undo,0,31,0,0,0,0,0,0,0,0,NULL
30 nj0 };
31 sK  static struct Gadget Gad={      /* Textgadget */
32 Od3    NULL,10,99,240,8,
33 Z4     GADGHCOMP,RELVERIFY,STRGADGET,
34 Rc     NULL,NULL,NULL,0,
35 LG     (APTR)&infoN,0,NULL
36 tp0 };
37 R3  static struct NewWindow FensterVereinbarung = { /* Fenster *
       /
38 e53    20,20,350,135,
39 ji     -1,-1,
40 4v     MOUSEBUTTONS| GADGETDOWN| GADGETUP,
41 kX     BORDERLESS| ACTIVATE,
42 z3     &Gad,NULL,NULL,NULL,NULL,
43 Hn     0,0,0,0,
44 Rw     WBENCHSCREEN,
45 2y0 };
46 1c  openw()
47 UR  /* Fenster und Lib's öffnen */
48 iB  {
49 nE3    if(!(IntuitionBase = OpenLibrary("intuition.library",0L))
          ) closew(-1);
50 IB     if(!(GfxBase = OpenLibrary("graphics.library",0L)))
          closew(-1);
51 JO     if(!(fenster = OpenWindow (&FensterVereinbarung)))
          closew(-1);
52 BV     RP1 =  fenster->RPort;
53 rM0 }
54 dM  closew(wert)
55 JY  /* Fenster und Lib's schließen */
56 f43    int wert;
57 rK0 {
58 ZP3    if(fenster)         CloseWindow(fenster);
59 AU     if(GfxBase)         CloseLibrary(GfxBase);
60 dp     if(IntuitionBase)  CloseLibrary(IntuitionBase);
61 yB8         if(first)       FreeMem(eintrag,sizeof(struct FileInf
            oBlock));
62 8r3    if(wert<0)           DisplayBeep(0);
63 1x     if(strlen(filename)==0) return(NULL);
64 SR     strcpy(Rwert,aktuellDir);
65 Tr     if(Rwert[strlen(Rwert)-1]!=':')   strcat(Rwert,"/");
66 vj     strcat(Rwert,filename);
67 Zq     return(Rwert);
68 6b0 }
69 9W  message()
70 GB  /* Nachricht, zum Beispiel Mausklick, empfangen */
71 5Y  {
72 7c4     int nklasse=0;
73 ry3    if (nachricht = GetMsg(fenster -> UserPort)){
74 JA6       nklasse = nachricht -> Class;
75 NH        ReplyMsg(nachricht);
76 Ej3    }
77 Y4     return(nklasse);
78 G10 }
79 pm  strmem(von)
80 gk  /* Platz schaffen für den eben gelesenen Namen */
81 FC3    char *von;
82 Gj0 {
83 KH3    char *nach;
84 1u     if((nach = AllocMem(strlen(von)+1,MEMF_PUBLIC | MEMF_CLEA
```

Listing 1. »Disk.h« dient zur komfortablen und sicheren Eingabe von Dateinamen. Bitte mit dem Checksummer (Seite 66) eingeben.

```
        R)) != NULL)
85 Xa7        strcpy(nach,von);
86 kj3    return(nach);
87 Pu0 }
88 eB  cd(newdir)
89 qg  /* festlegen einer neuen aktuellen Directory */
90 mV3    unsigned char *newdir;
91 Ps0 {
92 J13    int  i;
93 lK     nr=0;
94 Rw     strcpystore,aktuellDir); /* 'Sicherheitskopie' */
95 OL     if (*newdir == '/') {      /* aufsteigen */
96 sJ6       for (i=strlen(aktuellDir);aktuellDir[i]!='/' && aktuel
          lDir[i]!=':';i--);
97 VJ        if(aktuellDir[i] == ':') i++;
98 lm        aktuellDir[i] = '\0';
99 b63    }
100 ra    else if (newdir[strlen(newdir)-1] != ':'){ /* kein Rootdi
          rectory ? */
101 xo6      if(aktuellDir[strlen(aktuellDir)-1] != ':') strcat(akt
          uellDir,"/");
102 kl       strcat(aktuellDir, newdir); /* neues Verzeichnis anhän
          gen */
103 fA8       }
104 6I        else strcpy(aktuellDir, newdir); /* Rootdirectory */

105 96        if((dir = Lock(aktuellDir, ACCESS_READ)) != 0) UnLoc
          k(dir);
106 ud        else {    /* Das neue Verzeichnis existiert nicht ! *
          /
107 wr6      strcpy(aktuellDir,store);/* Rückkopieren der Sicherhei
          tskopie */
108 MV       return(1);   /* Fehler */
109 1G8       }
110 vU3    return(0);    /* OK */
111 nI0 }
112 WY  readDir()
113 lE  {
114 Kq3    int i;
115 JA     zaehler = 0;   /* zählt die Einträge */
116 cq     eintrag = (struct FileInfoBlock *) AllocMem(sizeof(struct
          FileInfoBlock),0);
117 Ve     dir = Lock(aktuellDir,ACCESS_READ);
118 Of     if((dir == NULL) | !Examine(dir,eintrag)) return(1);/* ni
          cht gefunden */
119 ID     if (first == 0){   /* Root-Directory lesen */
120 To6       first++;
121 HQ        if(!index(eintrag->fib_FileName,' ')){
122 2o9          strcpy(aktuellDir,eintrag->fib_FileName);
123 nI           strcat(aktuellDir,":");
124 OV6       }
125 1W3    }
126 PJ     ExNext(dir,eintrag);
127 2C     while (IoErr() != ERROR_NO_MORE_ENTRIES){ /* Dir-Einträge
           lesen */
128 n16       if(zaehler+1>80) continue;
129 QF        if (eintrag->fib_DirEntryType > 0){  /* Directoryname
          */
130 A59          zaehler++;
131 Gs           Ename[zaehler]=strmem(eintrag->fib_FileName);
132 V2           farbe[zaehler]=2;
133 9e6       }
134 zz        else{      /* Filename */
135 FA9          zaehler++;
136 Lx           Ename[zaehler]=strmem(eintrag->fib_FileName);
137 Y4           farbe[zaehler]=1;
138 Ej6       }
139 7m        ExNext(dir,eintrag); /* nächster Eintrag */
140 G13    }
141 2D8        for(i=zaehler+1;i<85;i++){    /* auffüllen mit Nullby
          tes */
142 4R6       strcpy(Ename[i],"");
143 8E        farbe[i]=0;
144 Kp8        }
145 U3         return(0);   /* OK */
146 Mr0 }
147 yD  print(text, col, xpos, ypos)
148 H9  /* Textausgabe (beliebige Farbe, beliebige Position) */
149 nT3    char *text;
150 q1     int cll, xpos, ypos;
151 Nq0 {
152 QX3    SetAPen(RP1, col);                /* Farbe wählen */
153 Oh     SetDrMd(RP1, JAM1);               /* Zeichenmodus */
154 kM     Move(RP1, xpos, ypos);            /* X/Y Position */
155 ZM     Text(RP1, text, strlen(text));    /* Text         */
156 W10 }
157 eP  block(col,x1,y1,x2,y2)
158 xs  /* Block zeichnen (beliebige Farbe, beliebige Position, beli
    ebige Größe */
159 fx3    int col,x1,x2,y1,y2;
160 Wz0 {
161 EZ3    SetAPen(RP1, col);                /* Farbe        */
162 a5     SetDrMd(RP1, JAM1);               /* Zeichenmodus */
163 Rf     RectFill(RP1, x1,y1,x2,y2);       /* Rechteck     */
164 e90 }
165 Yv  Hlight(lnr)
166 Xd  /* der gewählte Eintrag wird gekennzeichnet */
167 pJ3    int lnr;
168 e70 {
169 BK3    SetDrMd(RP1, COMPLEMENT);             /* Zeichenmodus */
170 bz     RectFill(RP1, 10,lnr+2,250,lnr+9); /* Rechteck      */
171 1G0 }
172 OW  setgad()
173 JC  {
174 Io3    int i;
175 5e     nr=0;
176 Kv     block(2,10,5,350,135);
177 TI     block(1,0,0,340,130); /* Block im Fenster */
178 ox     block(2,8,4,275,92);
179 Dg     block(0,10,5,250,91); /* großer Block */
180 bB     Wdir();              /* Schieber     */
181 B5     block(2,8,96,252,109);
182 ZO     block(0,10,97.250,108); /* Textgadget */
183 Kz     for(i=0;i<2;i++){
184 JD6       block(2,i*170+8,114,i*170+82,126);
185 Mc        block(0,i*170+10,115,i*170+80,125); /* ok & stop */
186 OV3    }
187 wa     print("OK",1,35,123);
188 IZ     print("STOP",1,199,123);
189 FV     print("A",2,295,110);
190 uX     print("G",2,300,115);
191 UA     for(i=0;i<3;i++){
192 fm6       block(2,283,i*33+4,332,i*33+26);
193 9u        block(0,285,i*33+5,330,i*33+25); /* 'df0:', 'df1:', 'c
          d /' */
194 ay        print(bef[i],1,292,i*33+17);
195 9e3    }
196 Af0 }
197 Hi  checkgad()
198 8b  /* Auswertung bei Mausklick */
199 9c  {
200 IE3    int i,x,y,knopfy;
201 zK     x=fenster->MouseX;
202 yv     y=fenster->MouseY; /* Position des Mauszeigers beim KLICK
        */
203 Ux     if(x>285 && x<330){
204 Kb6       for(i=0;i<3;i++){ /* 'df0:', 'df1:', 'cd /' */
205 Lv9          if(y>i*33+5 && y<i*33+25) return(i+1);
206 Kp6       }
207 Lq3    }
208 Mv     if(y>115 && y<125){
209 LI6       if(x> 10 && x< 80) return(4); /*  OK  */
210 zT        if(x>180 && x<250) return(5); /* STOP */
211 Pu3    }
212 Fa     if(x>254 && x<271 && y>5 && y<91){
       /* Schieberführung */
213 HX6       if(y<nr+6 && nr>0) return(6);                 /* über
          Knopf */
214 sS        if(ynnr+9 && nr<zaehler-10) return(7);       /* unter
          Knopf */
215 cy        if(y>=nr+6 && y<=nr+9){  /* Knopf */
216 Ky9          while((class=message()) != MOUSEBUTTONS){
217 DVC             knopfy=y;
218 9A              while((y=fenster->MouseY)==knopfy){
219 fxE               if((class=message())==MOUSEBUTTONS) return(0);

220 Y3C             }
221 CB              nr+=(y-knopfy);
222 B0              if(nr<0 || nr>zaehler-10){ /* zu hoch/tief */
223 MNF                nr-=(y-knopfy);
224 UH                 MAUSKLICK
225 61                 return(0);
226 e9C             }
227 AR              Wdir(); /* Directory neu schreiben */
```

Listing 1. »Disk.h« dient zur komfortablen und sicheren Eingabe von Dateinamen (Fortsetzung)

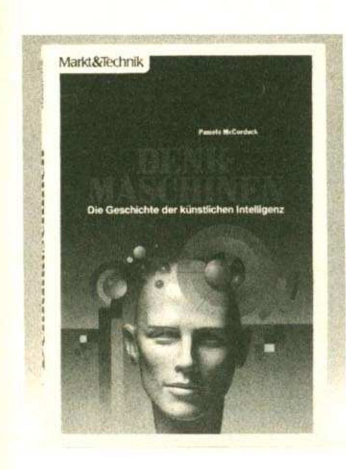
```
228 gB9          }
22  hC6        }
230 B6         return(0);
231 JE3      }
232 c9       if(x>10 && x<250 && y>8 && y<88){ /* Klick im großen
             Block */
233 pT6        fnr=nr+(y-2)/8;           /* Filenummer ermitteln  */
234 lo         return(8);
235 nI3      }
236 HC       return(0);
237 pK0  }
238 VH   fehler()
239 qc   /* Ein Verzeichnis wurde nicht gefunden */
240 oH   {
241 D73      int i,j;
242 o1       for(i=0;i<32;i++) block(0,301-i,111-i/2,301+i,111+i/2);
243 At       print("READ",2,286,107);
244 Fz       print("READ",3,285,108);
245 dw       print("ERROR",2,282,117);
246 Jy       print("ERROR",3,281,118);
247 lh       MAUSKLICK  /* warten auf Bestätigung durch Mausklick */
248 sf       MAUSKIICK
249 RJ       block(1,270,95,332,127);
250 EU       print("A",2,295,110);
251 tW       print("G",2,300,115);
252 4Z0  }
253 Xe   Wdir()
254 lR   /* Directoryeinträge in großen Block schreiben */
255 3W   {
256 o83      int i;
257 PV       block(0,10,5,250,91);           /* großer  Block  */
258 dL       block(0,252,5,273,91);          /* Schieberführung */
259 VF       block(3,254,nr+6,271,nr+9);     /* Schieber (Knopf) */
260 tp       if(zaehler==0){         /* kein Eintrag (bei Fehler) */
261 RD6        strcpy(filename,"");
262 U6         RefreshGadgets(&Gad,fenster,NULL);
263 Gp         return;
264 G13      }
265 10       for(i=1;i<=10;i++){ /* neu schreiben */
266 PZ6        print(Ename[nr+i],farbe[nr+i],10,(i+1)*8);
267 Jo3      }
268 Kp0  }
269 P9   Ndir(new)
270 zn3      char *new;
271 Jm0  {
272 ZK3      if(cd(new)) fehler(); /* neue Directory */
273 cF       if(readDir()){         /* Directory lesen */
274 GT6        fehler();
275 4M         zaehler=0;
276 Sx3      }
277 Eu       Wdir();                /* Directory schreiben */
278 Uz0  }
279 h9   disk()
280 Sv   {
281 Q13      int gnr, wert;
282 nT       int ende=0;
283 bL       openw();             /* Fenster und Lib's öffnen */
284 Zb       setgad();            /* Arbeitsfeld erstellen    */
285 ys       Ndir("SYS:");        /* erste Directory          */
286 7F       while(ende==0){
287 nV6        class=message(); /* Nachricht lesen    */
288 ZD         switch(class){
289 J09          case GADGETDOWN:  /* Textgadget übernimmt Intuition
                 */
290 qb           case  GADGETUP :{
291 clC            break;
292 iD9          }
293 ks           case MOUSEBUTTONS:{ /* Mausklick - mein Job */
294 EpC            gnr=checkgad(); /* 'Gadget' ermitteln */
295 pn             switch(gnr){
296 6pF              case 1:{ /* DF0: 'Gadget' */
297 9rI                first=0;
298 zE                 Ndir("DF0:");
299 hU                 MAUSKLICK
300 lu                 break;
301 rMF              }
302 K0               case 2:{ /* DF1: 'Gadget' */
303 FxI                first=0;
304 AQ                 Ndir("DF1:");
305 na                 MAUSKLICK
306 r0                 break;
307 xSF              }
308 En               case 3:{ /* CD / 'Gadget' */
309 7eI                Ndir("/");
310 sf                 MAUSKLICK
311 w5                 break;
312 2XF              }
313 B6               case 4:{ /* OK 'Gadget' */
314 wyI                wert=1;
315 1F                 ende++;
316 1A                 break;
317 7cF              }
318 00               case 5:{ /* STOP 'Gadget' */
319 N9I                strcpy(filename,"");
320 01                 wert=0;
321 rL                 ende++;
322 7G                 break;
323 D1F              }
324 aq               case 6:{ /* über Kopf */
325 EUI                nr--;
326 1Z                 Wdir();
327 9w                 MAUSKLICK
328 DM                 break;
329 JoF              }
330 hK               case 7:{ /* unter Knopf */
331 2EI                nr++;
332 7f                 Wdir();
333 F2                 MAUSKLICK
334 JS                 break;
335 PuF              }
336 r0               case 8:{ /* Knopf getroffen */
337 eZI                Hlight((fnr-nr)*8);
338 tF                 if(farbe[fnr]==1){
339 caL                  strcpy(filename,Ename[fnr]);
340 kM                   RefreshGadgets(&Gad,fenster,NULL);
341 NA                   MAUSKLICK
342 je                   Hlight((fnr-nr)*8);
343 Sb                   break;
344 Y3I                }
345 RE                 MAUSKLICK
346 oa                 strcpy(filename,"");
347 rT                 RefreshGadgets(&Gad,fenster,NULL);
348 pk                 Hlight((fnr-nr)*8);
349 zo                 Ndir(Ename[fnr]);
350 Zi                 break;
351 fAF              }
352 j6               default:{
353 clI                break;
354 iDF              }
355 wTC            } /* switch gnr */
356 fo             break;
357 Zd9          } /* case MOUSEBUTTONS */
358 pC           default:{
359 irD            break;
360 oJ9          }
361 TT6        } /* switch class */
362 sh3      } /* while */
363 MW       closew(wert);
364 sN0  }
(C) 1987 M&T
```

Listing 1. »Disk.h« dient zur komfortablen und sicheren Eingabe von Dateinamen (Schluß)

Programmname:	Disk.h
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.40
Aufrufe:	siehe Text

```
Programm : Demodisk
-------------------
 1 us0  #include <libraries/dosextens.h>
 2 2G   #include "disk.h"
 3 k6   main()
 4 0T   {
 5 ZL3      char *filename;
 6 Y2       if((filename=disk()) != NULL) Execute(filename,0L,0L);
 7 7c0  }
(C) 1987 M&T
```

Listing 2. Das Demoprogramm zu »Disk.h«. Es werden nur zwei Zeilen benötigt.



Denkmaschinen

»Die Künstliche Intelligenz (KI) ist eine Faszination« — jedenfalls wird man diesen Eindruck nicht mehr los, wenn man sich erst einmal in das Buch »Denkmaschinen« eingelesen hat. Hinter diesem Titel verbirgt sich kein Lehrbuch über irgendeine Programmiersprache, über irgendein Fachgebiet der Künstlichen Intelligenz, sondern die Autorin Pamela McCorduck stellt vielmehr die Geschichte der KI vor (so lautet auch der Untertitel des Buches). Beginnend mit dem Kapitel »Die Anfänge«, welches wirklich von der Basis der menschlichen und technischen Entwicklung von der Antike bis in die Neuzeit aus versucht, die Entstehung der KI darzustellen, werden eine Vielzahl an Beispielen, wissenschaftlichen Texten und geschichtlichen Fakten auf ansprechende Weise dargebracht. Dabei ist immer wieder zu erkennen, wie gründlich Pamela McCorduck recherchiert hat, um all die Aspekte der Entwicklung der KI zu beleuchten. In weiteren Teilen ihres Buches geht die Autorin auf die KI des 20. Jahrhunderts ein, beschreibt anhand zahlreicher Text- und Literaturausschnitte sowie vieler persönlicher Interviews die Lebens- und Arbeitsweise führender Größen der KI-Forschung. Darunter finden sich (neben vielen anderen) Namen wie Hubert Dreyfus, Edward Feigenbaum, Joseph Weizenbaum und John McCarthy (der Vater der Programmiersprache LISP). Die Ausführungen beschränken sich jedoch nicht nur in einem Lobgesang auf die KI. Pamela McCorduck zeigt auch die Widerstände gegen die KI, ihre Begründung und teilweise auch Widerlegung auf. Selbstverständlich werden auch Anwendungen der KI beleuchtet und unter dem Titel »Verwirklichung« näher dargebracht. Hier findet der interessierte Leser Themen wie Robotik, Projekte der KI und angewandte KI. Diese Themen wie auch der Rest des Buches sind anschaulich dargestellt. Das Buch ist für diejenigen zu empfehlen, die Spaß am Lesen und an der Auseinandersetzung mit der Geschichte der Informatik (beziehungsweise KI) haben. Die Autorin versteht es, auf eindrucksvolle Art und Weise Fakten, interessante Interviews und spannende geschichtliche Dokumente sowie fast schon philosophische Aussagen (oder ist KI etwa keine Philosophie?) zu kombinieren und auf eine den Leser ansprechende Weise darzubieten. Wer sich also nicht nur mit dem abfinden will, was er als »bekannt« vorgesetzt bekommt, sondern selbst einen Ausflug in Geschichte und Anwendung der KI machen möchte, der ist mit diesem Buch bestens bedient.

(Ingolf Krüger/pa)

Pamela McCorduck, Denkmaschinen, Markt & Technik Verlag AG, 345 Seiten, ISBN 3-89090-419-x, Preis: 49 Mark

Das große Amiga 2000 Buch

Das vorliegende Buch beschäftigt sich mit den Vorzügen und Leistungsmerkmalen des Amiga 2000 und dem damit verbundenen Grundlagenwissen. Nach einem kurzen Überblick über den Lieferumfang und die Anschlußmöglichkeiten des Amiga 2000 geben die Autoren eine Anleitung für den praktischen Umgang mit der Benutzeroberfläche und den zugehörigen Hilfsprogrammen. Das Werk ist in drei größere Themenkomplexe unterteilt. Im ersten Teil erlernt der Anfänger anhand einfacher und praxisorientierter Beispiele, wie man den Computer startet, und die mitgelieferten Programme sinnvoll einsetzt. Der zweite Teil stellt die bisher erhältlichen Programme vor und befaßt sich mit Tips und Tricks zur Installation von Amiga-Software auf einer Festplatte. Der dritte und umfangreichste Abschnitt des Buches enthält schließlich Informationen über die zur Zeit erhältlichen Erweiterungskarten und Hardware-Bausätze.

»Das große Amiga 2000 Buch« kann als Ergänzung der zum System gelieferten Dokumentation nur empfohlen werden. Sei es der Einbau von zusätzlichen Laufwerken oder die PC-Emulation, die Formatierung und das Partionieren der Festplatte, die Installation des Kickstarts im RAM, die Audio-Ausgänge für den PC oder sogar ein Absturz der Festplatte — die Autoren vermitteln wertvolle Tips für die Bewältigung solcher und anderer Probleme.

Ein Ausblick auf Erweiterungen, die in der Zukunft erwartet werden und eine ausführliche Vorstellung von Anwendungsprogrammen runden das informative Werk ab.

(Markus Zietlow/pa)

Christian Spanik/Hannes Rügheimer, Das große Amiga 2000 Buch, Data Becker GmbH, ISBN 3-89011-199-8, Preis: 59 Mark



Softwareführer '88

Der 700 Seiten umfassende Softwareführer im Taschenbuchformat enthält Beschreibungen und Kurzdaten von über 3000 Programmen. Schwerpunkt sind mit etwa 450 Seiten branchenneutrale (Textverarbeitung, Finanzbuchhaltung, Lohn und Gehalt, Adreß- und Dateiverwaltung) und branchenspezifische Programme (Handel, Handwerk, Immobilien, Dienstleistungen). Allerdings führen die Verfasser nur ein einziges Programm auf, das unter Amiga-DOS läuft. Die meisten Programme benötigen einen IBM-PC-kompatiblen Computer. Auf den ersten Blick ist das Buch also nur für die Anwender des Amiga 2000 mit PC-Schnittstelle interessant. Aber auch der an kommerzieller oder technisch-wissenschaftlicher Software interessierte Besitzer der anderen Amiga-Modelle kann sich mit dem Werk einen Überblick über den aktuellen Stand der Softwareentwicklung verschaffen.

Softwareführer '88, Dr. L. Rossipaul Verlagsg. m. b. H., Bavariaring 24, 8000 München 2, ISBN 3-87686-032-6, Preis: 34 Mark

Kommentiertes ROM-Listing

Mit dem zweiten Band des ROM-Listings enthüllt Dr. Ruprecht einen weiteren Teil des Amiga-Betriebssystems in Form eines kommentierten Assembler-Quellcode-Listings (Teil 1 haben wir in Ausgabe 11/87 vorgestellt). Diesmal geht es um »devices« und »resources«. Unter einem Device kann man sich zunächst ein Gerät oder eine Vorrichtung vorstellen, während eine Resource eher eine »Hilfsquelle« oder ein »Rohstoff« ist. Im Amiga-Betriebssystem sind Devices und Resources Sammlungen von Routinen und Daten, die der Computer für die Kommunikation des Systems mit der Außenwelt benötigt. Die Resources werden als untergeordnete Module sowohl von den Devices als auch anderen Systemroutinen genutzt. Bis auf das »audio.device« sind alle im Kickstart-ROM enthaltenen Devices (keyboard, gameport, input, console, timer, trackdisk, audio) und Resources (keymap, cia, potgo, disk, misc) aufgelistet. In einer Einführung beschreibt der Autor den Aufbau und die Arbeitsweise des im ROM enthaltenen »Kerns« sowie der im RAM angelegten Datenstrukturen der einzelnen Routinen. Den weitaus größten Teil des Werkes umfaßt mit 300 Seiten das gut kommentierte Listing.

Leser mit nur wenig Kenntnissen über das Betriebssystem des Amiga werden für das Verständnis sowohl der Einführung als auch des Listings ergänzende Literatur benötigen. Der Assembler- oder Systemprogrammierer sollte »Das kommentierte ROM-Listing« allerdings in seine Bibliothek aufnehmen. (pa)

Dr. Ruprecht, Kommentiertes ROM-Listing, Mediscript-Verlag, 340 Seiten, ISBN 3-88320-169-3, Preis: 69,— Mark



Amiga Speichererweiterung intern, Fastram
1 MB **749,—** 2 MB **1248,—** 4 MB **2048,—**
(Test A-Special 10/87)

Laufwerk 1036A, abschaltbar, anschlußfertig im amigafarbenen Blechgehäuse **329,—**
(Test Kickstart 10/87)

Amiga 500, 512 K abschaltbar **249,—**
Amiga 500, 2 MB abschaltbar **889,—**
Amiga Digitizer A500, 1000, 2000 **99,—**
Amiga Genlock-Interface **489,—**

Coll-Card und Coll-Prom, die ersten Epromkarten und Eprombrenner für den Amiga
Coll-Card A500 + A1000, 512 K best. **298,—**
weitere 512 K-Karte 49,— 2 MB **419,—**
Coll-Card A2000 2 MB **398,—**
Coll-Prom A500 + A1000 **398,—**
Coll-Prom A2000 **650,—**

Weiteres Zubehör (5¼"-Laufwerk, A500, Soft) auf Anfrage lieferbar. Anfragen ab 18.00 Uhr unter Telefon 06163/1278.

FUTURE VISION
Friedrich-Veith-Str. 21, 6128 Höchst/Odw.

Bekanntmachung!!
X-tension Harddisks
Made in Germany
20 MB 1598.—
40 MB 2098.—
72 MB 3475.—
Superschnell
durch Interleave 0
Schaltnetzteil u.v.m.

Fa. T. Eder & P. Archinger
Peyerstraße 35
8500 Nürnberg 80
0911/329558 17-20 Uhr

---

1 MByte-RAM-Karte für alle Bus-Systeme, z.B. VME, EC8 steckerfertig, 100x160 mm .... **548,— DM**
2 MByte-RAM-Karte für KWS 235x160 mm, steckerfertig .... **1098,— DM**
DIN-A3-Plotter CP64 für C64 .... **875,— DM**
CP65 für KWS oder andere Systeme (V.24/Centronics)
HP-GL-kompatible, 150 mm/sec.
Auflösung 0,05 mm .... **1111,— DM**
Komplett für KWS nur .... **1999,— DM**

41256-120	**6,40**	TTL-HC	
511000-12	**55,—**	00, 02, 04, 08,	
41464-120	**7,80**	10, 11, 20, 21	**0,60**
2764-250	**6,40**	73, 74, 86	**0,95**
27256-250	**9,80**	245	**2,20**
325572	**39,90**	HCT 04	**0,60**
ICL7109	**24,95**	137, 240, 241,	
MK50395N	**36,90**	244, 373, 374	**1,60**
MK50398N	**35,90**	C-MOS 4000, 01	
MAX232	**13,95**	02, 07, 11, 12	**0,45**
ICL7106	**9,40**	7805-7824	**0,95**
ICL7107	**9,40**	IC-Sockel	
XR2206	**8,—**	Low-cost pP	**0,018**
XR2207	**8,20**	IC-Sockel	
74LS03,10	**0,37**	Präz. pP	**0,04**

Händleranfragen erwünscht
Martin Fleitmann electronic
Albert-Schweitzer-Weg 12 · 4600 Dortmund 18
Telefon: 0231/673868

AMIGA-LAUFWERKE

5,25"-Laufwerk, 40/80 Track schaltbar und abschaltbar Frontblende und Metallgehäuse in Amigafarbe, komplett mit Turbocopy 429,- DM

3,5"-Slimline-Laufwerk (1036A) abschaltbar, Frontblende und Metallgehäuse in Amigafarbe, kompl. mit Turbocopy 349,- DM

3,5"-Slimline-Laufwerk für Amiga 2000 **intern**, bereits **modifiziert**, mit Einbauanleitung 245,- DM

VESALIA VERSAND G. DOES
MARIENWEG 40, 4230 WESEL
TELEFON 0281/65466



B.Ludewig + Th.Wittwer GbR
Bielefelder Str.121 · 4802 Halle (05201/7555

I.D.S.

3,5" Drive (NEC 1036 A) 340,–
!!! FLOPPYBUS DURCHGESCHLEIFT !!!
Metallgeh. / schaltbar / anschlußfertig

1 MByte CLTD (Amiga 10/87) 695,–
2 MByte Golem Box 895,–
(Beide auch für A-500)

CLTD Harddisks + Contr. bis 760 MB
〉〉〉〉〉〉〉 **20 MB 1499,–** 〈〈〈〈〈〈〈

!!! Soundsampler 99,– !!!
Genlock · Soundscape · Videodigi's
NEC, Star,OKI, Panasonic-Drucker

Rufen Sie an, lassen Sie sich beraten
07263/5693

★P★E★G★G★Y★

Bevor Sie sich irgendwelche Programme wie:

- **Copy-Tool**
- **Disk-Monitor**
- **File-Monitor**
- **System-Monitor**
- **CLI-Tool, Shell**
- **Tasten-Editor**
- **Drucker-Spooler**
- **Task-Handler**
- **Fullscreen-Editor**

oder sonst ein Werkzeug zur Programm-Entwicklung anschaffen, sollten Sie sich, **in Ihrem eigenen Interesse,** über *PEGGY* informieren.

PEGGY, die komfortable Programmier-Umgebung von der ersten Zeile bis zum fertigen Programm. Für alle Programmiersprachen geeignet!

PEGGY 2.12 **nur DM 128.–**
PEGGY's Editor incl. Demo nur **DM 10.–**

SAS-BERND
SYSTEM- & ANWENDERSOFTWARE
Langgasse 93 · 5216 Niederkassel 5
☎ (02 28) 45 26 26

PUBLIC DOMAIN
inkl. 2DD-Disks 3,–

für Amiga 500–1000–2000
z.B.: Fish, Faug, Panorama, Amicus, Auge, ACS, Amuse, SACC, Casa Mi Amiga...
Jede 3,5"-Disk kostet inkl. 2DD-Disks nur bei Abnahme von:
1–5 Disks 4,–, 6–10 Disks 3,70 DM, 11–19 Disks 3,50 DM, ab 20 Disks 3,–
Wir liefern ca. 95% aller erhältlichen Public Domain. Disks, die sich noch nicht in unserem PDV (Public-Domain-Verzeichnis) befinden, tauschen wir ein. Alle Disks einzeln nach freier Auswahl von jeder Serie (keine Mindestmenge).
Public-Domain-Verzeichnis auf 3,5"-Disks 5,– (bar, Briefmarken)
Neu!! Alle PDs werden auch auf Markendisks geliefert (Sentinel, SKC, Fuji), Aufpreis nur 1,50 DM

3,5"-Disketten No Name
2DD 2,28–2,80 DM

1DD 1,99–2,39 DM
Hierzu Staffel, Tagespreise telefonisch erfragen. Alle Disks in 10/50-Packs inkl. Aufkeber. 100% Error free.
Sentinel 3,5"-Disks + schwarze Hardbox: Superpreis
100 5,25" 2D No Name 79,– DM
Ausführl. Infomaterial gg. frankierten Rückumschlag!! Wir liefern per UPS-Nachnahme. Versandkosten insges. 8,– bei Vorauskasse (Scheck...) 5,– (Ausland 15,–/10,–)

Datentechnik M. Bittendorf
Postfach 100248, 6360 Friedberg
Telefon 06031/61950 (9–19 Uhr)



AMIGA: Speichererweiterung
für 512K zusätzliches RAM (Echtzeituhr nachrüstbar), Komplett mit 512K: **189.-**
Leerplatine + Stecker für AMIGA 500: **39.-**
Uhrenchip 6242 **24.-**
3,5" Laufwerke für Amiga/Atari ST
für Amiga anschlußfertig m. Gehäuse **298.-**
für Atari St w.o. und Netzteil **329.-**
TEAC FD135FN 1MB 2,7cm hoch **239.-**
BESTELLUNG/VERSAND
ALCOMP · A. Lanfermann
Lessingstr. 46 · 5012 Bedburg · Tel. 02272/1580

AMIGA-SOFTWARE

FASTLIGHTNING

Der neue Maßstab, kopiert eine Disk bei drei Disketten gleichzeitig, in **23 Sekunden**. Vier Kopiermodi bis 81 Tracks. Kopiert **fast** alle Formate und Kopierschutz 89,- DM

WHITE LIGHTNING mit Formatiermodus und Verify, sehr schnell und sicher. Das Meistbenutzte (Amiga Spez. 12/87) für nur ein Laufwerk 49,- DM

TURBOCOPY für zwei Laufwerke zum Erstellen von Sicherheitskopien, sehr schnell und sicher 59,- DM

VESALIA VERSAND G. DOES
MARIENWEG 40, 4230 WESEL
TELEFON 0281/65466

TV · VIDEO · HIFI · COMPUTER

HANS-JÜRGEN KRAPFL



6730 Neustadt
Am Knappengraben 25
Tel. 06321/80568



AMIGA UTILITIES

Genlock 8700 für alle AMIGA DM 1.059,-

Polaroid Palette KAMERASTATION mit Interface Imprint für alle AMIGA DM 6.750,-

DIGI VIEW DM 398,- RGB Splitter DM 598,-

Pro Video CGI Font Animation DM 498,-

"DISNEY" Apprentice: ANIMATION DM 698,-
Organisches Animationsprogramm jetzt in Hi-Res, dt. Handbuch

Videoscape 3D incl. dt. Handbuch DM 398,-

Sculpt 3D incl. dt. Handbuch DM 229,-

SILVER incl. dt Handbuch DM 349,-

deutsche Handbücher solo DM 39,95

LOFTPOST + 4 DEMO DISKS: 2 Apprentice, 1 Pro Video, 1 Videoscape 3D DM 20,-
im Brief (Schein o. Marken) o. KtNr. 61655 Stadtsparkasse Kassel BLZ 520 501 51 o. NN + DM 6,- NN geb.

video LOFT film HARD & SOFT GmbH
Fiedlerstr. 22-32
D-3500 Kassel
tel.: 0561-873399
fax: 0561-878048



Q-Line-Serie. Programme für den Atari ST, Amiga und PCs.
Exklusiv bei MEGA /// TEAM.
Fordern Sie unsere Unterlagen an.

MEGA /// TEAM

Bannizastr. 31, 4250 Bottrop
Telefon 02041/94842.

Minipreise für Amiga-Laufwerke

1 MB 3½" FD 35 FN
Teak, Metallgehäuse, anschlußfertig 310 DM

1 MB 5¼" FD 55 FR
Teak, Metallgehäuse, anschlußfertig, 80/40 Sp. 375 DM

1 MB 3½" FD 35 FN Teak 270 DM

1 MB 5¼" FD 55 FR Teak 290 DM

Angelika Huber · Elektr. Bedarf
Wörnitzstraße 3
8850 Donauwörth
Telefon 0906/5567





Tel.: 07307 - 6230

AMIGA 500/1000/2000

Terrorpods	65,-	Uninvited	69,-
Roadwar 2000	65,-	Impact	45,-
Knight Orc	59,-	Ogre	69,-
Firepower	69,-	Defender of the Crown	69,-
Deja Vu	79,-	Bad Cat	59,-
Demolition	25,-	Phantasie III	69,-
Feud	29,-		
Hunt for Red October	69,-	**ZUBEHÖR**	
Garrison	65,-		
Portal	79,-	Abdeckh. Weichplastik f. 500	20,-
Ninja Mission	29,-	dto. Tastatur 1000/2000	14,-
Gnome Ranger	45,-	dto. Systemeinheit/Mon. 1000	39,-
Tass Times i. Tone Town	69,-	dto. Systemeinheit/Mon. 2000	59,-
The Pawn	65,-	Disketten Multitronic 2DD	
Karate Kid II	65,-	10 St.	39,-

Katalog mit Programmbeschreibungen gegen DM 0,80 in Briefmarken (System unbedingt angeben!).
Versandkosten: bis 150,- **Vorauskasse** 3,50, **Nachnahme** 6,-, ab 150,- keine Versandkosten.
Achtung: Irrtümer, Preisänderungen und Streichungen bleiben vorbehalten.

Commodore Ihr Spezialist AMIGA
W.A.W.-ELEKTRONIK

Autorisierter Commodore Service & Fachhändler

Wir fangen an, wo andere aufhören!
Mit Beratung, Verkauf, Service & Betreuung!
Vom C64 bis zum Commodore Amiga & PC-AT

Unser Angebot:

Amiga 1000 Speichererweiterung auf 1 Megabyte (Fronteinbau nicht sichtbar) mit Anleitung 299,- DM
Amiga 2000 Harddisk (ohne PC-Karte)
20 MB incl. SCSI-CRT 1495,- DM
40 MB incl. SCSI-CRT 1995,- DM

Wir führen die gesamte Commodore-Palette, nicht zu Dumping-Preisen, sondern mit dem Service der für uns spricht.
Weiterhin OKIDATA-Computer-Drucker und Thomson Computer-Monitore, sowie Zubehör & Ersatzteile.
Literatur & Software von:



und diverser In- und ausländischer Anbieter

Liste geg. Freiumschlag! Bestellungen nur schriftlich.

W.A.W. Elektronik
Tegeler Straße 2 · 1000 Berlin 28
☎ **030/4043331**

Mo.-Fr. 10-13 und 15-18 Uhr
Sa. 10-14 Uhr

AMIGA-SOFTWARE ZU SUPERPREISEN

Fred-Fish .... **Nr. 1-102** FAUG .... **Nr. 1-39**
PANORAMA .... **Nr. 1- 48** AUGE 4000 .... **Nr. 1-12**
Amicus .... **Nr. 1- 16** und andere ....
(Katalogdisk gegen 5.- DM anfordern)
Einzeldisk .... **5.00 je Disk** ab 40 St. .... **4.30 je Disk**
ab 10 St. .... **4.80 je Disk** ab 100 St. .... **4.00 je Disk**
ab 20 St. .... **4.50 je Disk** alle 262 St. zu .. **999.00 DM**
alle Preise incl. 2 DD Diskette

TORNADOS - Die Super PD-Serie. Alle Prg. laufen garantiert auf Amiga 500/1000/2000.
Nr. 1-30 (incl. 2 DD Disk) **6.00 je Disk**
(Katalogdisk gegen 5.- DM anfordern)
BOOTWRITER - kostenlose Infos anfordern **17.- DM**
IFF-CON - kostenlose Infos anfordern **27.- DM**
Inland: Porto + Verpackung 3.- DM je Bestellung
Ausland: Porto + Verpackung 6.- DM je Bestellung
(nicht bei Anforderung v. Infos od. Katalogdisk)
Lieferung gegen Vorkasse oder V-Scheck
Achtung! Kein Ladenverkauf!
Bestellung und Anfragen an:

Bestellung und Anfragen an:
PD-Shop
Opladener Straße 30, D-4018 Langenfeld

Amiga Laufwerk NEC 1036 A
für Amiga 500/1000/2000
3,5 Zoll, slimline
(10,5x17,5x3,5 cm)
amigafarbener Strukturlack Einbrennlack.,
Metallgehäuse anschlußfertig
349,— DM

2 MB RAM Erweiterung
autokonfig, abschaltbar, durchgeführter Bus,
amigafarbenes Metallgehäuse
100% Software-kompatibel
929,— DM

SWS Computersysteme G.d.b.R.
Beratung — Planung — Verkauf — Service
Am Rathaus 8, 8395 Hauzenberg
Telefon 08586-2174

Modula-2 kontra C

C ist momentan die populärste Programmiersprache für den Amiga. Aber Modula steht in den Startlöchern, um der renommierten Sprache Paroli zu bieten. Kann Modula sich im Rennen behaupten?

In Ausgabe 6/7 haben wir einen Überblick über einen Modula-Compiler vermittelt (Seite 100, Ein genialer Schachzug!) und schnelle C-Compiler vorgestellt (Seite 90, Schnelle Compiler). In dieser Ausgabe gehen wir der Frage nach, wie sich das TDI-Modula (Version 3.00) gegenüber den getesteten C-Compilern behaupten kann.

Damit wollen wir dazu beitragen, die Diskussion über Vorzüge der einzelnen Sprachsysteme für den Amiga anzuregen und unseren Lesern Entscheidungshilfen anzubieten.

Um noch einmal daran zu erinnern: Es fanden zwei Diskettenlaufwerke Verwendung und für Zwischendateien wurde die RAM-Disk benutzt. Quelltexte wurden in die RAM-Disk geladen und von hier aus durch die Compiler/Assembler beziehungsweise Linker bearbeitet. Dadurch wurde gleichzeitig bewirkt, daß die lauffähigen Programme in der RAM-Disk abgelegt und von hier aus gestartet wurden. Um für den Test von Modula gleiche Bedingungen zu schaffen, ist in der RAM-Disk das Unterverzeichnis »T« anzulegen und »T:« auf »Ram:T« mit ASSIGN zu definieren.

Die Quelltexte der beiden erwähnten Testprogramme in C sollen nicht wiederholt werden. In Modula übertragen, lesen sie sich wie in Listing 1 und 2 dargestellt.

Überraschende Testergebnisse

Im Modul »NumbTest« (Listing 1) setzt sich »Feld« im Gegensatz zum Vergleichsmodul aus dem Basistyp »CARDINAL« zusammen, der positive ganze Zahlen auf 2 Byte abbildet. Die bedingte Kompatibilität der C-Datentypen »char« (1 Byte breit) und »int« (2 Byte breit) ist in Modula nicht gegeben. Sie ist explizit zu programmieren durch Voranstellen des jeweiligen Bezeichners (Beispiel: Z sei vom Typ »BYTE« und I vom Typ »INTEGER«). Es sind die folgenden Zuweisungen möglich:

```
Z := BYTE(I);
```

und

```
I := INTEGER(Z);)
```

Von dieser Möglichkeit wurde in dem Programm nicht Gebrauch gemacht, da die CPU gleich viel Zeit benötigt, um 1 oder 2 Byte breite Datentypen zu speichern. Auch in Modula kann der Sinus nur für eine Gleitkommazahl ermittelt werden. (Strenggenommen gehört die Sinusfunktion nicht zum Sprachumfang. Die mathematischen Funktion werden in sogenannten »MathLib« verfügbar gehalten.) Es stehen die Datentypen »REAL« und »LONGREAL« zur Verfügung, die 32 beziehungsweise 64 Bit einnehmen. Für jeden Datentyp ist eine eigene Bibliothek vorhanden. Um dem Testprogramm in C zu entsprechen, Sinuswerte mit einfacher Genauigkeit zu ermitteln und zu speichern, wurde in »NumberTest« der Datentyp »LONGREAL« verwendet, obwohl der Typ »REAL« zweckmäßiger gewesen wäre. Zusammengefaßt läßt sich feststellen, daß die Übertragung des Programms von C in Modula ohne Probleme möglich war.

Auch das zweite Beispiel, Listing 2, war ohne weiteres in Modula zu übernehmen, ohne daß die Lesbarkeit oder die Struktur darunter litten. Modula verfügt über keine elegante Ausdrucksform für In- beziehungsweise Dekrementierungen wie C, sie sind ausdrücklich zu programmieren (Beispiel siehe Listing 2, vierte Zeile von unten). In dem Modul wird die Prozedur »ADR« aus dem Pseudomodul »SYSTEM« importiert, mit der die Adresse einer definierten Variable ermittelt werden kann.

Die Sprache C gilt allgemein als schnell. Die festgestellten Unterschiede zwischen den beiden getesteten C-Compilern relativieren diese globale Aussage. Wie ist nun die TDI-Implementation von Modula einzuordnen? Tabelle 1 gibt möglicherweise eine überraschende Antwort:

NumbTest.Mod entspricht Test.C und Primes.Mod gleicht Primzahl.c. Länge des Compilats in Byte: Modula kann auf zwei Arten Linken, daher zwei Angaben, durch einen Schrägstrich getrennt.

Aufs einzelne Byte geschaut

Mit der TDI-Bibliothek »MathLib0« benötigt NumbTest lange 23 Sekunden, mit unserer Implementation »MathTransLib« hingegen nur knapp 15 Sekunden.

Aus Tabelle 1, 2 und 3 geht deutlich hervor, daß das Modula von TDI leistungsmäßig in der Nähe von Aztek-C einzuordnen ist. Hätten Sie das gedacht? Um das Ergebnis zu verallgemeinern, wären noch weitere Tests erforderlich. Ein Trend darf jedoch abgeleitet werden: Es ist unverkennbar, daß C nicht die einzige Hochsprache ist, mit der sich flotte Programme für den Amiga realisieren lassen. Auch die Summe der Zeiten für das Übersetzen und Binden kann sich sehen lassen. Die lauffähigen Programme sind außerdem recht kompakt.

In einem weiteren Schritt wurde die unterschiedliche Laufzeit der Programme ausschnittsweise und exemplarisch analysiert. Die beiden Spitzenreiter (Aztek-C und Modula) mußten dazu herhalten. Es wurde die erste Schleife der Programme Test.c beziehungsweise TestNumb.Mod gewählt. Die Compiler erzeugen Codes, die Sie in Listing 3 und 4 sehen können.

```
MODULE NumbTest; (*$Q+,$T-*)
(* Q+ : JSR in BSR ersetzen, um Vorteile des kleinen Speichermo-
dells in C z. T. zu kompensieren; T- : Bereichsprüfung für Felder Ab-
schalten, um C im Verhalten zu simulieren *)
FROM MathLib0 IMPORT sin;
FROM Terminal IMPORT WriteLn, WriteString;
CONST max  = 32000;
VAR   Feld  : ARRAY[0..max] OF CARDINAL;
        i, l  : CARDINAL;
        z     : LONGREAL;
BEGIN
  WriteString("INTEGER"); WriteLn;
  FOR i := 0 TO 10000 DO
     l := 3 * i;
  END;
  WriteString("DOUBLE"); WriteLn;
  FOR i := 0 TO 10000 DO
        z := sin(FLOAT(i)) * 3.0;
  END;
  WriteString("FELD"); WriteLn;
  FOR i := 0 TO max DO
        Feld[i] := max - iö
  END;
  WriteString("FERTIG");
END NumbTest.
```

Listing 1. Transzendentale Funktionen kosten Zeit

```
MODULE Primes; (*$Q+,$T-*)
FROM SYSTEM IMPORT ADR;
VAR PrimZahlen          : ARRAY[0..3500] OF CARDINAL;
    PrimZahlenZeiger    : POINTER TO CARDINAL;
    Zahl, Teil          : CARDINAL;
BEGIN
   PrimZahlenZeiger := ADR(PrimZahlen);
   FOR Zahl := 5 TO 32000 DO
        Teil := 3;
        WHILE ((Teil * Teil)<=Zahl) AND ((Zahl MOD Teil)<>0) DO
          INC(Teil, 2);
        END;
        IF (Teil * Teil) > Zahl THEN
          PrimZahlenZeiger^ := Zahl;
          INC(PrimZahlenZeiger, 2);
        END;
   END;
END Primes.
```

Listing 2. Beliebter Benchmarktest mit Primzahlen

	Aztek-C 3.40	TDI-Modula 3.00	Lattice-C 3.10
Compilieren			
NumbTest	18	20	30
Primes	14	17	24
Linken			
NumbTest	46	25/19	49
Primes	26	7/7	40
Länge des Compilats			
NumbTest	6352	3048/8380	16528
Primes	1928	1280/4036	5772
Laufzeit			
NumbTest	13	23/15	51
Primes	13	14	27

Tabelle 1. Aztec-C und TDI-Modula nahezu gleich »stark«

Die eigentliche FOR-Schleife fängt in beiden Codeausschnitten jeweils in der zweiten Zeile an. In der ersten Zeile wird die Variable »i« auf den Wert 0 gesetzt. Dabei fällt auf, daß Aztek die Variable indirekt mit Distanz adressiert, während TDI die absolute lange Adressierung vorzieht. Der Sachverhalt wäre nicht erwähnenswert, wenn die Adressierungsarten zeitneutral blieben. Die unterschiedlichen Techniken der Adressierung bewirken auch, daß der TDI-Code länger ausfällt. Der Block mit den Nullen in Listing 4 sticht ins Auge. Beide Codeausschnitte umfassen sieben Instruktionen mit 28 beziehungsweise 36 Byte. Der Zeitbedarf für einen Durchlauf der Schleife beträgt 134 (Aztek) beziehungsweise 160 (TDI) Zyklen. Der Unterschied von 26 Zyklen entspricht einem zusätzlichen Zeitverbrauch von gut 3,6 µs für einen Schleifendurchlauf. Bei 10000 Schleifendurchläufen benötigt das TDI-Modul 260000 zusätzliche Zyklen,

Verschiedene Adressierung

entsprechend 36,31 Millisekunden. Der zusätzliche Zeitbedarf für die betrachtete Schleife könnte durchaus auf 60000 Zeilen linear zu durchlaufenden Assemblercode hochgerechnet werden. Die Dimension des mit Aztek-C erreichbaren Zeitvorteils wird nun vollends deutlich.

Die Ausführungen zur FOR-Schleife fanden ihre Bestätigung in dem Code der WHILE-Schleife des Moduls »Primes.Mod« (Listing 2), beziehungsweise im entsprechenden C-Programm. Die WHILE-Schleife konsumiert 370 (TDI-Modula) beziehungsweise 342 (Aztek-C) Zyklen. Die zusätzlichen Zyklen gehen ausschließlich auf das Konto der angewandten Adressierung.

```
426d fffe           clr.w     -2(a5)              i = 0
362d fffe           move.w    -2(a5),d3
c6fc 0003           mulu      #3,d3               i*3 -->d3
3b43 fffc           move.w    d3,-4(a5)           d3 --> 1
526d fffe           add.w     #1,-2(a5)           i = i + 1
0c6d 2710 fffe      cmp.w     #10000,-2(a5) i     = 10000?
6de8                blt       ffe8
```

Listing 3. Eine FOR-Schleife in C benötigt 134 Taktzyklen

```
4279 0000 FA02           CLR.W     0000FA02            i = 0
7A03                     MOVEQ     #0003,D5
CAF9 0000 FA02           MULU      0000FA02,D5         i*3 -->d5
33C5 0000 FA04           MOVE.W    D5,0000FA04         d5 --> 1
5279 0000 FA02           ADDQ.W    #0001,0000FA02      i = i + 1
0C79 2711 0000 Fa02 CMPI.W    #2711,0000FA02      i = 10000?
66E2                     BNE       FFE2
```

Listing 4. FOR-Schleife in TDI-Modula mit 160 Taktzyklen

Der unterschiedliche Zeitbedarf der analysierten Programme ist aber noch in einem anderen Zusammenhang zu sehen. Der Aztek-Compiler kann mit zwei unterschiedlichen Speichermodellen arbeiten, dem großen und dem kleinen. Nur bei Anwendung des kleinen Modells, das für den Test benutzt wurde, können die vorteilhaften Adressierungsarten eingesetzt werden. Bei Programmen mit einer Länge von über 64 KByte ist im allgemeinen das große Speichermodell einzusetzen, das häufig die absolute lange Adressierung mit 4 Byte erzwingt. Die Geschwindigkeitsvorteile des Aztek-Compilers gegenüber dem TDI-Compiler verringern sich somit bei langen Programmen.

Bei Vergleich der Textausgabe (Listing 3 und 4) fällt auf, daß in Modula Zeilenwechsel und Sprung zum Zeilenanfang mit »WriteLn« explizit zu programmieren sind. Daraus folgt, daß für die Ausgabe einer Textzeile in Modula zwei Prozeduraufrufe (WriteString und WriteLn) erforderlich sind. In C besorgt dieses elegant die Prozedur »printf«. Prozeduraufrufe kosten Zeit. In unserem Beispiel wirkt sich dieser Sachverhalt jedoch praktisch nicht aus.

Es reizte, den hohen Unterschied in den Laufzeiten der Programme Test (Aztek) und NumbTest (TDI) zu ergründen. An der FOR-Schleife lag es ja nicht, wie wir bereits wissen. Die Ursache — Sie ahnen es schon — wurde in der unterschiedlichen Realisierung der Sinusfunktion vermutet. Die Analyse der Bibliotheken der beiden Programmiersysteme ergab, daß Aztek intern Gleitpunktzahlen im Fast Floating Point-Format (FFPs) von Motorola darstellt und unter anderem die schnellen transzendentalen Routinen (ebenfalls von Motorola) nutzt, die dem Betriebssystem des Amiga beigegeben sind. TDI hingegen bildet Gleitpunktzahlen im zukunftsweisenden IEEE-Format (Standard 754 des »Institute of Electrical and Electronics Engineers, Inc.«) ab und setzt eigene transzendentale Routinen ein, die in Hochsprache programmiert sind, sowie etwas umständlich wirken. Vermutlich kam es TDI darauf an, ein einziges Modul für alle Standardbibliotheken zu realisieren, das auf beliebiger Hardware läuft, um den Entwicklungsaufwand gering zu halten. Ein gutes Beispiel für die Portabilität mit erheblichen Nachteilen für den Anwender.

Mit den aufgezeigten Nachteilen müssen Modula-Anwender jedoch nicht länger leben. Wir haben einen einfachen Weg gefunden, auch von Modula aus auf die speicherresidenten FFPs zuzugreifen. Bei Einsatz unserer Sinusroutine benötigt das Programm NumbTest nur eine gute Sekunde länger als Test (Aztek-C).

Unterschiede in der Textausgabe

Es wird häufig der Sprache C eine besonders hohe Eignung zugesprochen, Systemroutinen des Amiga aufzurufen, weil das Betriebssystem eben in dieser Sprache geschrieben ist. Diese Annahme klingt zunächst plausibel, verliert aber bei Analyse der tatsächlichen Vorgänge an Gewicht. Bei Aufruf von Systemprozeduren in C und in Modula wird generell wie folgt verfahren: Das Anwenderprogramm übergibt beim Aufruf einer Schnittstellenprozedur Parameter. Diese lädt die Parameter in bestimmte Register und ruft die entsprechende Systemprozedur auf.

```
                    Aztek-C                     TDI-Modula
Aufruf einer Prozedur:
Quelltext: Draw(rp, 10, 20);              Draw(P, 10, 20);
-------------------------------------------------------------
Assembler: move.w   #20,-(a7)             MOVE.L P,-(A7)
           move.w   #10,-(a7)             MOVE.W #000A,-(A7)
           move.l   _rp,-(a7)             MOVE.W #0014,-(A7)
           jsr      _Draw                 JSR    Draw         (XX.L)
           add.w    #08,a7                ADDQ.L #0008,A7
-------------------------------------------------------------

Die aufgerufene Schnittstellenprozedur zum Betriebssystem:

                                          LINK   A6,#0000
                                          MOVE.L A6,-(A7)
           move.l   4(sp),a1              MOVE.L 000C(A6),A1
           movem.l  8(sp),d0/d1           MOVE.W 000A(A6),D0
                                          MOVE.W 0008(A6),D1
           move.l   _GfxBase,a6           MOVE.L GraphBase,A6
           jmp      -246(a6)              JSR    FFOA(A6)
                                          MOVE.L (A7)+,A6
                                          UNLK   A6
                                          RTS
```

Listing 5. Aztek C und TDI Modula — 11 Mikrosekunden Unterschied beim Aufruf von Systemroutinen

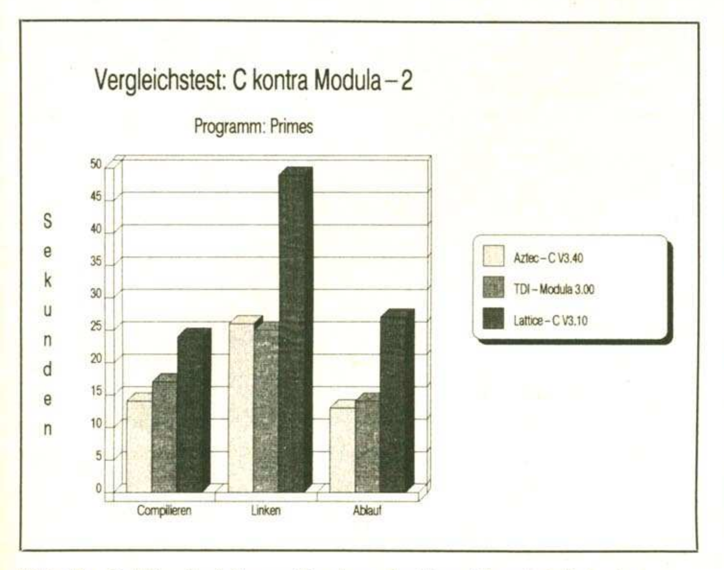


Tabelle 2. Die drei Compiler im direkten Vergleich beim Primzahlenprogramm »Primes«

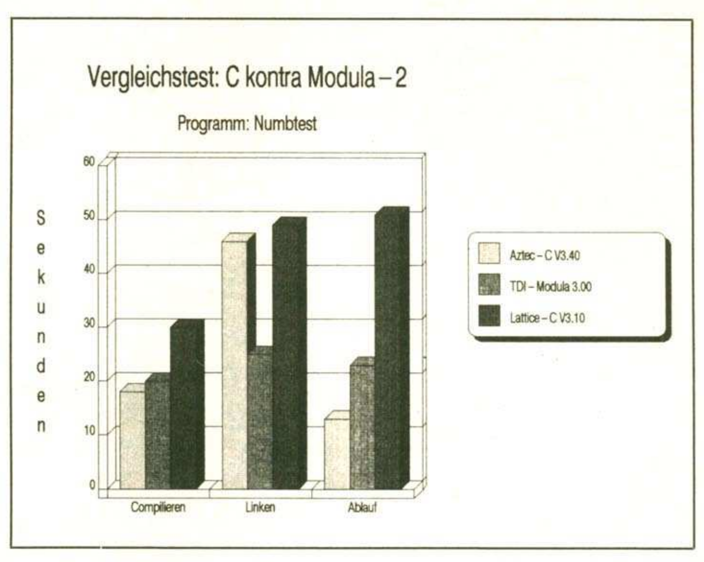


Tabelle 3. Auch bei dem zweiten Benchmarktest zeigen sich die Unterschiede zwischen den drei Compilern

Die Programmausschnitte in Listing 5 zeigen im ersten Teil, daß beide Compiler die gleiche Technik der Parameterübergabe bei Aufruf von Prozeduren verwenden.

Um die Schnittstellenprozedur aufzurufen, erzeugt der Aztek-Compiler einen Code, der sich mit 72 Zyklen begnügt, während der TDI-Compiler 80 benötigt, also ungefähr eine Mikrosekunde länger. Aus den Codefragmenten geht auch hervor, daß keiner der Compiler direkt auf die Systemroutine »Draw« zugreift. Beide Compiler versorgen die eigentliche Schnittstellenprozedur mit den zu verwendenden Parametern über den Stack.

Die eigentliche Schnittstellenprozedur fällt jedoch in C und in Modula unterschiedlich aus. Der TDI-Compiler verpackt den Aufruf der Systemroutine in den üblichen Eintritts- und Austrittscode einer Prozedur. Der Aztek-Compiler hingegen integriert eine »vorfabrizierte« Assemblerroutine und erzeugt zudem noch einen raffinierten, zeitlich kurzen Rücksprung zum aufrufenden Programm. Es ist somit nicht verwunderlich, daß die Zeitbilanz zugunsten von Aztek-C ausfällt. TDI-Modula benötigt für den eigentlichen Aufruf von Draw 138 Zyklen, während Aztek-C mit 66 auskommt.

Insgesamt gesehen ergibt sich, daß TDI-Modula für den Aufruf von Draw 80 Zyklen, das heißt zirka elf Mikrosekunden, mehr benötigt als Aztek-C. Die zeitliche Differenz sollte man jedoch für die Praxis nicht überbewerten. Die Ausführungen lassen auch erkennen, wo durch eigene Programmierung anzusetzen ist, um den Ablauf von Modula-Programmen zu beschleunigen.

C hat einen Konkurrenten

Unsere Ausführungen weisen nach, daß das Modula von TDI es Wert ist, in den Club der schnellen Compiler aufgenommen zu werden. Möglicherweise haben wir etwas an dem Sockel gerüttelt, auf dem C-Compiler häufig plaziert werden. Mit TDI-Modula steht ein Compiler in der Leistungsklasse des Aztek-C zum Preis des Lattice-C zur Verfügung. Es muß also nicht unbedingt C sein, um auf dem Amiga schnelle Programme zu realisieren. Außerdem ist Modula prädestiniert, moderne Konzepte der Softwaretechnologie zu realisieren, die uns Programmierern das Leben erleichtern können.

Es wurde auch deutlich, daß moderne, offene Programmiersysteme nicht mehr so einfach vergleichbar sind, wie ehemals zum Beispiel Interpreter mit festem Sprachumfang. Compiler können, wie in unserem Beispiel, durchaus gleiche Leistungen aufweisen, die durch ungünstig angelegte Bibliotheken verzerrt werden.

(Edgar Meyzis/rb)

C und Assembler – ein starkes Paar

Viele Programmierer meinen, auch auf dem Amiga in Assembler das letzte aus dem Computer holen zu müssen. Aber der erhöhte Aufwand und die größere Fehleranfälligkeit sprechen dagegen. Ein besserer Weg ist das Einbinden von Assemblerteilen in ein C-Programm, das den Gesamtablauf steuert.

Der Amiga ist ein schneller Computer und es besteht selten die Notwendigkeit aus Geschwindigkeitsgründen ein Programm komplett in Assembler zu schreiben. Oder halten Sie es für nötig, daß ein Programm zehnmal so schnell auf Ihre Eingabe wartet?

Ein einfacher Weg auch große Programme mit möglichst geringem Aufwand und hoher Effektivität zu schreiben, ist das Verwenden einzelner Assemblerroutinen in einem C-Programm. In C sind viele Dinge wesentlich einfacher zu erledigen als in Assembler, wobei der Geschwindigkeitsverlust zu vernachlässigen ist. Ein Beispiel sind die Ausgabefunktionen. Es ist wirklich sehr einfach und sicher in C eine Datei zu öffnen und Daten hineinzuschreiben oder herauszulesen. In Assembler sind hier schon wesentlich mehr Fehlerquellen vorhanden, da der Quelltext viel länger ist. Außerdem unterscheidet sich die Geschwindigkeit kaum. Es gibt viele Beispiele bei denen außer mehr Arbeit beim Programmieren kein Unterschied zu erkennen ist.



Illustration: Rolf Boyke

Übersichtlich und schnell

Darum sollten Sie in Zukunft den einfacheren Weg gehen. Zuerst schreiben Sie Ihr Programm in C, ungeachtet Ihrer Bedenken in bezug auf Geschwindigkeit. Danach müssen Sie feststellen, welche Programmteile besonders oft aufgerufen werden. Durch Umsetzen dieser Teile in Assembler erhalten Sie im Verhältnis zum Aufwand eine spürbare Beschleunigung.

Auf die Frage wie man feststellt wie oft einzelne Funktionen benutzt werden, gibt es eine sehr einfache Antwort. Auf der Public Domain-Diskette der Panorama-Serie gibt es ein Programm, durch das Sie diese Informationen erhalten. Auf der Diskette »Panorama 14d« befinden sich im Unterverzeichnis »Profiler« die zwei Programme »p1« und »p2«. Um sie zu benutzen, müssen Sie allerdings den Quellcode oder zumindest die Objektdatei des Programms besitzen. Sie erhalten nach dem Programmlauf dann eine genaue Liste wie oft welche Funktion verwendet und wie lange das gedauert hat.

Die Besitzer eines Aztec-C-Compilers haben einen sehr großen Vorteil, sie können Assemblercode direkt in ihr C-Programm integrieren. Dies geschieht mit der Präprozessor-Anweisung »#asm«. Nach dieser Anweisung folgt die Assemblerroutine, die benötigt wird. Zum Abschluß dient dann der Befehl »#endasm«. Von der Verwendung solcher Assemblerteile merkt man beim Übersetzen natürlich nichts. Es wird nämlich nur folgendes gemacht: der Assemblerteil wird unverändert in den Assemblercode eingefügt, der aus den C-Anweisungen generiert wird. Es erfolgt dabei keine Makroersetzung. Eine bestimmte Konstruktion muß allerdings vermieden werden. Wenn Sie nach einer if-Abfrage einen Assemblerteil einbauen wollen, muß nach dem if-Teil eine leere Anweisung folgen. Es muß also folgendermaßen aussehen:

```
if (...)
{
   ...
}
;
#asm
 ...
#endasm
```

Wenn Sie den Strichpunkt vergessen, arbeitet das Programm bei nicht erfüllter if-Bedingung nach der #endasm-Anweisung weiter.

Ansonsten gibt es aber keine weiteren Probleme. Wer zum Beispiel auf Variablen des C-Programms zugreifen will, hat eine fantastisch einfache Möglichkeit. Die benötigten Variablen müssen global definiert sein, dann können Sie mit dem Variablennamen und einem Unterstrich davor auf die Veränderliche zugreifen. Man kann aber auch den Weg gehen, den auch C benutzt: die

Übergabe über den Stack. Ein Beispielprogramm zur schnellen Ausgabe eines Speicherbereiches sehen Sie im Listing. »ZeichenSatz« ist dabei ein Zeiger auf die Punktmatrix der benötigten Zeichen. Die Datei mit den Bytes, die die Zeichen ergeben, erzeugen Sie am besten mit Basic. Dazu schreiben Sie den gewünschten Buchstaben auf den Bildschirm und holen ihn mit der GET-Anweisung (für Grafik) in eine Feldvariable und speichern die so erhaltenen Werte in eine Datei. Es werden die Ziffern (0 bis 9), die Buchstaben (A bis F) und zwei weitere Zeichen (». « und »: «) benötigt. Natürlich können Sie auch die Werte selbst ausrechnen und in die Datei schreiben oder direkt im C-Programm unterbringen. Der Zeiger auf die Bitmap »BM« ist auch global definiert, wodurch der Zugriff recht einfach ist. Die Adresse (»adr«), ab der angezeigt werden soll, wird wie von C gewohnt übergeben.

Wenn Sie sich nicht sehr gut mit der Arbeitsweise des C-Compilers auskennen, sollten Sie innerhalb einer Funktion nicht abwechselnd C- und Assembler-Anweisungen benutzen. Abgesehen von der schlechten Lesbarkeit des Programmes ist die Veränderung der vom Compiler verwendeten Register eine gute Möglichkeit, den Guru aus seinem Versteck hervorzulocken.

Eine bewährte Methode ist die Definition einer eigenen Funktion für den Assemblercode (Listing). Dabei ist jedoch folgendes zu beachten: Der Aztek-C-Compiler generiert zu Beginn der Funktion eine Link-Anweisung. Mit dieser Anweisung wird in der Regel Platz für lokale Variablen auf dem Stack reserviert. Wenn nun keine lokale Variablen definiert werden (wie in unserem Beispiel), setzt der Compiler

```
LINK A5,0
```

ein. Damit wird zwar kein Platz auf dem Stapel reserviert, aber der Inhalt des Registers A5 dort abgelegt. Das bedeutet, daß eventuell über den Stack übergebene Daten für die Assemblerroutinen sich nicht direkt oberhalb der Rückkehradresse, sondern 4 Byte höher befinden.

Die zweite Konsequenz dieser Arbeitsweise ist die Plazierung einer UNLINK-Anweisung durch den Compiler am Ende der Funktion. Diese muß auf jeden Fall ausgeführt werden und deswegen darf die »Assemblerfunktion« nicht mit RTS enden. Wenn die Ausführung der Assemblerroutine nicht am Schluß der Funktion beendet ist, muß ein Sprung an dieses Ende eingesetzt werden (Listing).

Die Benutzer des Lattice-C-Compilers haben die oben beschriebenen Möglichkeiten leider nicht. Dabei muß jedoch auf einiges geachtet werden. In einer der nächsten Ausgaben wird dieses Thema ausführlich behandelt werden.

Was Sie bis jetzt gesehen haben, hat Sie sicher davon überzeugt, daß es sehr selten sinnvoll ist, ein Programm komplett in Assembler zu schreiben. Mit der Verwendung von Assemblerunterroutinen in C-Programmen können Sie genausogut alles erreichen, was Sie anstreben.

(rb/pa)

Programmname:	Memshow
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.40
Aufrufe:	cc Memshow +l -s ln Memshow.o -lc32

```
Programm : Memshow
------------------

  1 AL0 #include <stdio.h>
  2 Qf  #include <intuition/intuition.h>
  3 K8  #include <intuition/intuitionbase.h>
  4 bo  #include <graphics/rastport.h>
  5 Li  char  ZeichenSatz[200];
  6 jO  long int BM;
  7 KB  struct Window *win;
  8 Qg  struct RastPort *rp;
  9 c5  struct IntuitionBase *IntuitionBase;
 10 g8  struct Screen *Screen;
 11 xa  struct GfxBase *GfxBase;
 12 4Y  struct BitMap *BitMap;
 13 TY  void ausgabe(adr)
 14 OM  long int adr;
 15 Be  {
 16 4k  #asm
 17 gV  AnzahlZeile EQU 20
 18 Oy  BytesProZeile EQU 16
 19 rR  ZeilenProZeichen EQU 8
 20 z5  BildAusgabe:
 21 Kx2   MOVEM.l  A0-A6/D0-D7,-(A7)
 22 Vb    MOVE.L 68(A7),A5 ; Startadresse
 23 60    MOVE.l  _BM,A6 ; PlanePointer
 24 LB    LEA  _ZeichenSatz,A3 ; Zeichensatz
 25 4J    ADD.L  #2002,A6
 26 cj    MOVEQ #80,d3
 27 rz    MOVE.L #AnzahlZeile-1,d4
 28 qW0 BALoop:
 29 XG2   MOVE.L A6,A4
 30 uE    BSR  ZeilenAusgabe
 31 jP    ADD.L  #80*ZeilenProZeichen,A6
 32 uu    DBRA  D4,BALoop
 33 Ei0 test:
 34 qF1  MOVEM.l  (a7)+,A0-A6/D0-D7
 35 852   BRA  Ende
 36 Js0 ZeilenAusgabe:
 37 UB2   MOVE.L A5,D0
 38 qA    BSR   LongOut
 39 TE    BSR  DpOut
 40 Vo    MOVE.L #BytesProZeile-1,D5
 41 pt0 ZALoop:
 42 xR2   ADD.L  #1,A4
 43 bZ    MOVE.B (A5)+,D0
 44 FZ    BSR  ByteOut
 45 Vy    DBRA  D5,ZALoop
 46 Ad    RTS
 47 nI0 LongOut:
 48 wC2   MOVE.L #7,D1
 49 uN    BRA  B01
 50 En0 ByteOut:
 51 Vf2   MOVE.L #1,D1
 52 vo    ASL.l  #8,D0
 53 wp    ASL.l  #8,D0
 54 xq    ASL.l  #8,D0
 55 SS0 B01:
 56 gN2   MOVE.L D0,D2
 57 zt0 BOLoop:
 58 Pm2   MOVEQ #3,D6
 59 Rk    CLR.L  D0
 60 nz0 BO2Loop:
 61 VN2   ASL.L  #1,D2
 62 Ur    ROXL.L #1,D0
 63 xa    DBRA  D6,BO2Loop
 64 XS    BSR  NibbleOut
 65 hs    DBRA  D1,BOLoop
 66 Ux    RTS
 67 yy0 DpOut:
 68 9g2   MOVEQ #16,D0
 69 LP0 NibbleOut:
 70 tX2   MOVE.L A4,A1
 71 jb    ASL.L  #3,D0
 72 tX    MOVE.L A3,A2
 73 PN    ADD.L  D0,A2
 74 xP    MOVEQ #7,D7
 75 f10 NOLoop:
 76 DF2   MOVE.B (A2)+,(A1)
 77 ff    ADD.L  D3,11
 78 Sx    DBRA  D7,NOLoop
 79 Ia    ADDQ.L  #1,A4
 80 iB    RTS
 81 k80 Ende:
 82 ys  #endasm
 83 Lq  }
 84 q8  void main()
 85 Jm  {
 86 Uu3    FILE  *fp;
 87 ey     char  *datei,c;
 88 Fe     int   i;
 89 wE     register long int adr;
 90 4s     if(!(GfxBase = (struct GfxBase *)
 91 m46       OpenLibrary("graphics.library",0L)))
 92 Qt        {
 93 Sm9          printf("\f\n\n\t\tNo Graphics");
 94 N3           exit(20);
 95 X26       }
 96 E83    if(!(IntuitionBase = (struct IntuitionBase *)
 97 vJ6       OpenLibrary("intuition.library",0L)))
 98 Wz        {
 99 2f9          printf("\f\n\n\t\tNo Intuition");
100 T9           exit(20);
101 d86       }
102 Du3    printf("\f\n\tDateiname: ");
103 Gv     scanf("%s",datei);
104 ij     if((fp=fopen(datei,"r"))==NULL)
105 d6     {
106 la6       printf("\f\n\tKann Datei %s nicht öffnen!\n");
107 cJ        exit(30);
108 kF3    }
109 8v     for(i=0;i<=17*8;i++)
110 T16       ZeichenSatz[i]=fgetc(fp);
111 Vd3    BM= IntuitionBase -> ActiveWindow ->
           RPort -> BitMap->Planes[0];
112 Fa     adr=0;
113 xd     for (adr=0;adr<16384;adr+=16)
114 3d6       ausgabe(adr);
115 zr3    fclose(fp);
116 Io     CloseLibrary(GfxBase);
117 Aj     CloseLibrary(IntuitionBase);
118 uP0 }
(C) 1987 M&T
```

Listing. Ein Demoprogramm in C, das mit Hilfe einer Assemblerunterroutine den Speicherinhalt schnell auf den Bildschirm ausgibt. Bitte mit dem Checksummer (Seite 66) eingeben.

Ein interessanter Fis(c)hzug

Bei Analyse der Public Domain-Diskette 53 aus der Fred Fish-Sammlung fiel uns im Unterverzeichnis »Compiler« ein C-Übersetzer mit dazugehörigem Quelltext auf. Wir haben uns damit näher befaßt, um Ihnen darüber zu berichten.

Der Compiler trägt die Bezeichnung »cc68k« und umfaßt zirka 78 KByte. Er arbeitet überraschend schnell und erzeugt Assemblerquelltext im Format »Unix 68000«. Dieses Zwischenergebnis ist noch mit einem Editor (zum Beispiel ED), einem Assembler (zum Beispiel Seka oder Devpac) und anschließend mit einem Linker (zum Beispiel Blink) zu bearbeiten, um ein lauffähiges Programm zu erhalten. Assembler und Linker befinden sich nicht auf der Diskette. Es soll jedoch entsprechende Programme im Bereich der Public Domain-Software geben, die Fred Fish im Begleittext (»read.me.fnf«) anspricht. Zum Beispiel finden Sie den Linker »Blink«, der auch von Lattice verwendet wird, auf den Fred Fish-Disketten mit den Nummern 34, 40 und 76.

Um den Compiler sinnvoll einsetzen zu können, benötigt man also weitere Programmierwerkzeuge und einiges an »Know-how«. Aber beides kann man auf den Public Domain-Disketten finden.

Die Diskette enthält mehrere Programmbeispiele. Wir haben »sieve.c« durch den Compiler übersetzen lassen. Es wurde, wie bereits erwähnt, Quelltext in einem besonderen Assemblerformat in der Datei »sieve.s« abgelegt. Darüber hinaus wurde eine Referenzdatei (»sieve.lis«) erzeugt, die den Quelltext in C mit Zeilennummern sowie eine Symboltabelle enthielt.

Es gelang uns nach manueller Nachbehandlung des Assemblerquelltextes (»sieve.s«) das Beispielprogramm auch über die Hürde des Assemblierens zu bringen. Dazu war das Laufzeitmodul »lib.s« einzubinden. Die Nachbehandlung beschränkte sich im wesentlichen auf das Umwandeln beziehungsweise Entfernen von Zeichen (»%« und »&«), die unter Unix eine andere Bedeutung haben als in der Amiga-Umgebung. Auch andere Texte müssen noch gelöscht oder ersetzt werden. Ein kurzes Stück Assemblercode, das von cc68k erzeugt wurde, sehen Sie im Bild. Die Arbeit war mit einem Editor schnell getan. Assemblerkenntnisse waren dabei von Vorteil. Nach dem Assemblieren hatten wir endlich ein Zwischenergebnis, das wie gewohnt weiter verarbeitet werden konnte. Es bereitete keinerlei Mühe, den vom Assembler erzeugten Code zu linken (zusammenbinden). Das so erstellte Programm lief einwandfrei. Zugegeben, der Weg vom Quelltext zum lauffähigen Programm war langwierig, es hat aber Spaß gemacht. An dem Beispiel wird deutlich, welchen Schwierigkeiten man sich beim Einsatz des »cc68k« aussetzt.

Der Autor des Compilers (Mathew Brandt) stellt die Einschränkungen seines Übersetzers in der Datei »compiler/read.me« selbst dar. Der Compiler gilt als unvollendet (bitte nicht mit einem musikalischen Werk verwechseln). Das Laufzeitsystem ist äußerst schwach ausgelegt. Der Autor erwähnt natürlich nicht, daß der Compiler nur ganz wenige Routinen des Betriebssystems des Amiga einsetzt. Mit anderen Worten, Sound und Grafik sowie alle anderen hervorragenden Eigenschaften des Amiga werden nicht unterstützt. Der Übersetzer war ja ursprünglich auch nicht für dieses System konzipiert. Weiterhin ist zu erwähnen, daß der Codegenerator Gleitpunktzahlen nicht verarbeiten kann, der Präprozessor begrenzte Funktionen hat, der Umfang an Datentypen begrenzt ist, und Probleme im Umgang mit Funktionen auftreten. Fred Fish läßt in seinem Begleitbrief auch Zweifel aufkommen, ob der Compiler C-Quelltexte auch wirklich richtig interpretiert und umsetzt. Unsere Experimente verliefen jedenfalls zufriedenstellend. Für echte C-Freaks stellt der Compiler aber einen wirklichen Leckerbissen dar, wie wir im folgenden noch sehen werden.

Zuerst wollten wir gar nicht glauben, den gesamten Quelltext (über 320 KByte) zu einem lauffähigen Compiler vor uns zu haben. Schnell wurde die Vollständigkeit anhand der Dateien »Makefile« und »link.cmd« überprüft. Tatsächlich, es lag uns die Version komplett vor, die Fred Fish durch einen Lattice-Compiler geschickt hatte, um »cc68k« zu erzeugen. Und nun begannen viele Stunden Studium der einzelnen Programmteile. Wir wollten doch schon immer einmal einem Compilerbauer über die Schulter schauen, um zu verstehen, wie ein C-Compiler angelegt ist. Die zum Teil gut dokumentierten Programmodule haben uns viele Fragen beantwortet. Besonders interessant waren die Module »analyze.c«, »getsym.c«, »preproc.c« und »gencode.c«. Es wurde schnell deutlich, daß es keines hohen Aufwandes bedarf, um zum Beispiel den Präprozessor leistungsfähiger zu gestalten sowie Quelltext erzeugen zu lassen, den auf dem Amiga verwendbare Assembler ohne Vorbehandlung verstehen. Viele Ideen stellten sich ein. So wäre es zum Beispiel möglich, den Compiler auf Animation zu spezialisieren. Kurzum, der Quelltext des Compilers ist hervorragend als Plattform geeignet, auf der weiterführende Arbeiten aufsetzen können, zum Beispiel die Schaffung eines Public Domain-Programmiersystems für den Amiga.

Zusammenfassend ist festzustellen, daß der C-Compiler in seiner jetzigen Form als Lernsystem wenig geeignet erscheint, weil die Einbettung in eine vollständige Programmierumgebung und der Zugriff auf wesentliche Betriebssystemroutinen des Amiga fehlen sowie nur einfache C-Routinen übersetzt werden. Für denjenigen, der jedoch die recht hohen Kosten für einen kommerziellen C-Compiler scheut, aber schon Erfahrung mit dem Amiga besitzt, eine sehr interessante Möglichkeit, in die Programmiersprache C hineinzuschnuppern.

Der Compiler ist für Profis insofern attraktiv, daß die Quelltexte tiefe Einblicke in die Praxis des Compilerbaus vermitteln können. Sie sind auch als Basis für ein selbstgeschriebenes Programmiersystem geeignet. Zum Schluß noch eine Warnung: Fassen Sie die Diskette erst an, wenn Ihr Terminkalender nur noch freie Seiten ausweist!

(Edgar Meyzis/rb)

```
        text
main:
        link    %a6,&-8
        movm.l  &0x1000,-(%a7)
        mov     &0,%d3
        mov.l   &1,-8(%a6)
        mov.l   %d3,-(%a7)
        jsr     subr
        add.w   &4,%a7
L%0:
        movm.l  (%a7)+,&0x0008
        unlk    %a6
        rts
subr:
        link    %a6,&0
        movm.l  &0x1000,-(%a7)
        mov.l   8(%a6),%d3
        mov     &2,%d0
        muls    %d3,%d0
        mov.l   %d0,%d3
        mov.l   %d3,%d0
L%1:
        movm.l  (%a7)+,&0x0008
        unlk    %a6
        rts
        global  main
        global  subr
```

Bild. Ein Ausschnitt aus einem von cc68k erzeugten Assemblercode

Lisp contra Prolog

Als die Sprachen der Künstlichen Intelligenz sind Lisp und Prolog bekannt. Welche Vorteile haben die beiden gegenüber Basic oder C? Lohnt sich der Umstieg?

Wenn Sie sich als Basic-Programmierer ein Lisp- oder Prolog-Programm anschauen, so werden Sie nur wenig Bekanntes wiederfinden. Selbst für Programmierer mit Kenntnissen von C, Pascal oder Modula sind solche Programme zuweilen undurchschaubar. Die Ursache hierfür sind die Erfordernisse, die das hauptsächliche Anwendungsgebiet — die Künstliche Intelligenz — an eine Programmiersprache stellt. Für das Verständnis dieser Einzelheiten sind grundlegende Kenntnisse über Aufgaben und Arbeitsweise der »KI« sinnvoll. Unternehmen wir also einen kleinen Ausflug in eine Wissenschaft, welche die Dummheit eines Computers zu verbergen sucht.

Der Begriff »KI« ist die Abkürzung für »Künstliche Intelligenz« und bezeichnet den Versuch, menschliches Denken auf einen Computer zu übertragen (Tabelle 1). Das derzeit wichtigste Anwendungsgebiet der KI sind (Programm-)Systeme, die Fachleute auf ihrem Arbeitsgebiet unterstützen oder (in Zukunft) sogar ersetzen können. Solche Systeme heißen »Expertensysteme«. Expertensysteme sind Programme, die Fachleute bei der Lösung eines Problems »simulieren« sollen.

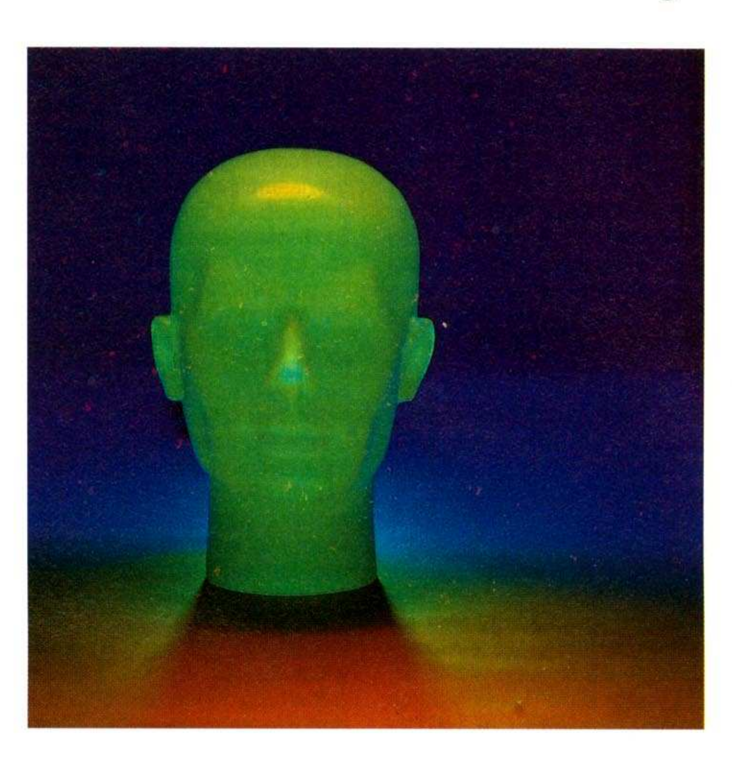
Expertenwissen im Speicher

Dies geschieht durch die logische Verknüpfung des in ihnen gespeicherten Wissens. Die Gesamtheit von Fakten und Regeln eines Expertensystems bezeichnet man als Wissens- oder Datenbasis. Ein kleines Beispiel soll dies verdeutlichen:

Für ein Expertensystem aus dem Bereich der Tierkunde könnte eine sehr einfache (modellhafte) Wissensbasis folgendes Aussehen haben:

Nahrung (Pferd,Pflanzen).
Nahrung (Hund,Fleisch).

Diese Fakten sind zu lesen wie: »Die Nahrung eines Pferdes sind Pflanzen« und »Die Nahrung eines Hundes ist Fleisch«. Um »Fragen« an das System stellen zu können, müssen nun noch Regeln festgelegt werden, die eine Verbindung zwischen den einzelnen Fakten herstellen. Regeln sind



Aussagen der Form »Wenn (Bedingung erfüllt), dann ...« Auf unser Beispiel angewandt, könnte eine einfache Regel lauten: »Wenn das Tier Pflanzen als Nahrung hat, dann ist es Vegetarier.« Eine mögliche Frage an unser Expertensystem wäre also: Welches Tier ist Vegetarier?

Um Fragen dieser Art korrekt zu beantworten, bedarf es außer einer Wissensbasis auch noch eines Schlußfolgerungsmechanismus (Inferenzmechanismus), der die Fakten anhand der zur Verfügung stehenden Regeln verknüpft und die zur Beantwortung der Fragen notwendigen Schlüsse daraus zu ziehen vermag.

Da die Arbeit an einem Expertensystem immer interaktiv stattfindet (im Dialog zwischen Mensch und Computer), steht auch hierfür eine eigene Komponente des Systems bereit, welche die Schnittstelle zwischen Computer und Benutzer bildet: das Dialogsystem. Das Dialogsystem hat je nach Art des Expertensystems unterschiedliches Aussehen. So kann der Kontakt zum Computer über eine grafische Benutzeroberfläche (zum Beispiel im Bereich der Chemie: Analyse von Molekularstrukturen) oder rein textorientiert (medizinische Anwendungen: Diagnose von aufgetretenen Symptomen beziehungsweise Ermittlung geeigneter Therapien) geschehen. Das Dialogsystem nimmt also die Eingaben des Benutzers entgegen und stellt eventuelle Antworten des Systems auf dem Bildschirm dar.

Die »Erklärungskomponente«, die eng mit dem Inferenzmechanismus und dem Dialogsystem zusammenarbeitet, begründet dem Anwender die Schlüsse des Expertensystems und den bisher eingeschlagenen Lösungsweg. Über die Erklärungskomponente teilt das Expertensystem dem Benutzer (neben der eigentlichen Beantwortung der Frage) beispielsweise mit: »Ich halte das Pferd für einen Vegetarier, weil es Pflanzen als Nahrung verwendet.« Dadurch kann der Anwender die Glaubwürdigkeit des Ergebnisses überprüfen. Das ist notwendig, da selbst das »beste« Expertensystem nie alle Eventualitäten berücksichtigen kann (sei es wegen Informationsmängeln oder Fehleingaben).

Sehr gute Expertensysteme verfügen über die Fähigkeit zu »lernen«, das heißt, aus bisherigen Schlüssen oder mit der Unterstützung eines »menschlichen« Experten, ihre Wissensbasis zu erweitern.

Anhand unseres obigen Beispiels können Sie sich selbst ein Bild von der Arbeitsweise des wichtigsten Teils im Expertensystem, der Inferenzmaschine, machen. Betrachten Sie dazu zunächst die Frage und überlegen Sie, auf welche Weise Sie selbst zum richtigen Ergebnis gelangen. Das Expertensystem beziehungsweise der Inferenzmechanismus geht dabei nicht anders vor als Sie es vermutlich getan haben. Zunächst werden die in der Frage auftauchenden Schlüsselwörter in der Wissensbasis gesucht, um die Frage überhaupt beantworten zu können. Danach werden die bekannten Fakten in die Regel eingesetzt und das Ergebnis ermittelt. Dieser Vorgang, in unserem Beispiel noch überschaubar und leicht nachzuvollziehen, ist in Wirklichkeit allerdings weitaus komplizierter.

Häufig werden Expertensysteme mit herkömmlichen Datenbanken verglichen oder sogar mit ihnen gleichgesetzt.

Mehr als eine Datenbank

Doch Expertensysteme sind wesentlich leistungsfähiger als Datenbanken. Ein Hauptunterschied zwischen einer Datenbank und einem Expertensystem ist die Tatsache, daß Datenbanken lediglich der Sammlung und Verwaltung von Daten dienen. Die Interpretation der ein- beziehungsweise ausgegebenen Daten obliegt noch immer dem Anwender. Im Gegensatz dazu verknüpfen Expertensysteme Daten und Regeln über den Schlußfolgerungsprozeß (Inferenzmaschine), stellen neue Regeln auf und erweitern ihre Wissensbasis. Schließlich wird das Ergebnis (welches durch den Schlußfolgerungsprozeß und nicht durch einen einfachen Suchalgorithmus ermittelt wurde) dem Benutzer präsentiert (Dialogsystem und Erklärungskomponente).

Zur Programmierung solcher »intelligenten« Systeme bedarf es grundsätzlich neuer Programmiersprachen, die Konzepte bereithalten, welche der Erstellung wissens- beziehungsweise symbolorientierter Programme entgegenkommen. Die zwei bekanntesten sind Lisp und Prolog. Obwohl für den Amiga bisher nur der LISP-Interpreter von MetaComCo erhältlich ist, kann in absehbarer Zeit auch mit einem Prolog-Compiler gerechnet werden.

Eine erste lauffähige Version von Lisp entstand gegen Ende der 50er Jahre. Lisp gilt als die zweitälteste Computerhochsprache. Sie wurde von John McCarthy am MIT (Massachusetts Institute of Technology) entwickelt und wird bis heute ständig erweitert. Inzwischen haben sich jedoch zum Beispiel mit »CommonLisp« und »CambridgeLisp« Standards etablieren können.

Lisp (LISt Processing) ist eine Programmiersprache, die ursprünglich nur zur Verarbeitung von Listenstrukturen entwickelt wurde. Das Listenkonzept von Lisp erwies sich jedoch als sehr geeignet zur Erstellung symbol- und wissensorientierter Programme (zum Beispiel Expertensysteme).

Listen einmal anders

Doch was ist unter dem Begriff »Liste« zu verstehen? Lisp enthält im Gegensatz zu anderen Programmiersprachen nur zwei Datentypen. Einer dieser Typen ist das »Atom«, der andere die »Liste«. Als Atom wird die kleinste und damit unteilbare Dateneinheit von Lisp bezeichnet. Man unterscheidet zwischen »numeric«-Atomen, also Atomen, welche Zahlen repräsentieren und »literal«-Atomen, die der Verarbeitung von Zeichen beziehungsweise Zeichenketten dienen. Man kann also sagen, daß Atome ähnlichen Zwecken dienen, wie die Datentypen einer konventionellen Programmiersprache (Integerzahlen, Zeichen- oder Stringvariablen).

Eine Liste besteht aus Atomen und/oder weiteren Listen. Listen sind mit Feldern oder verketteten Recordstrukturen anderer Programmiersprachen vergleichbar. Verkettete Recordstrukturen (wie sie in Pascal, C und Modula-2 benutzt werden) lassen sich wohl am besten mit Listen vergleichen, da Felder meist typgebunden sind und diese Einschränkung auf Listen gerade nicht zutrifft.

Lisp wurde von vornherein so konzipiert, daß ein größeres Problem durch Aufteilung in kleinere Funktionen zerlegt (modularisiert) werden kann. Die Sprache verzichtet dabei auf herkömmliche Prozeduren und verwendet ausschließlich Funktionen. Solche Funktionen sind vergleichbar mit den »functions« in C und Pascal, beziehungsweise mit den »Funktionsprozeduren« in Modula-2. Eine Funktion gibt im Gegensatz zu einer Prozedur ein Resultat zurück.

Ein weitaus gravierender Unterschied zwischen Lisp und konventionellen Programmiersprachen ist das fast völlige Fehlen sequentieller und iterativer Kontrollstrukturen. Das bedeutet, daß der Lisp-Programmierer auf die üblichen Schleifenkonstruktionen wie »FOR..NEXT«, »REPEAT..UNTIL« und »WHILE..DO« verzichten muß. An ihre Stelle treten rekursive Funktionen. Eine Funktion heißt dann »rekursiv«, wenn sie sich selbst aufruft. Anders als »REPEAT..UNTIL«, das eine Anweisungsfolge solange ausführt, bis eine bestimmte Bedingung erfüllt ist, ruft sich eine Lisp-Funktion (welche die REPEAT..UNTIL-Schleife ersetzen soll) solange selbst auf, bis auch hier eine Bedingung erfüllt ist. Bild 1 zeigt die Umsetzung der Fakultätsfunktion in Lisp.

Dieser auf den ersten Blick völlig undurchsichtige Dschungel von Klammern und Buchstaben stellt die Definition einer Funktion in Lisp dar. Die Funktion heißt »FAK« und hat mit »n« nur einen Parameter. Im Verlauf der Funktion wird die Bedingung (CONDition) »n=0« überprüft (zerop n). Ist die Bedingung wahr, so ist die letzte Rekursionsebene erreicht und »1« wird als Wert zurückgegeben. Im andern Fall wird FAK in der Anweisung »(times n (FAK(sub1 n)))« — das ist der Schritt n*FAK(n-1) — aufgerufen. Das Interessante an diesem Beispiel ist einerseits die Ausführungsgeschwindigkeit, mit der ein Lisp-Interpreter die Berechnung vornimmt und andererseits die Rechengenauigkeit. Der Interpreter bricht die Berechnung nicht nach einer gewissen Genauigkeit ab, sondern gibt das Ergebnis in voller Stellenzahl aus. (Dieses Beispielprogramm wurde mit dem Lisp-Interpreter/Compiler von Metacomco getestet.)

```
(de FAK (n)
 (cond ((zerop n) 1)
    ( t (times n (FAK (sub1 n))))))
```

Bild 1. Lisp berechnet die Fakultät

Prolog

Prolog ist, verglichen mit Lisp, eine noch relativ »junge« Programmiersprache. Sie wurde um 1972 von Alain Colmerauer an der Universität von Marseille entwickelt. Der Name ist eine Abkürzung für »PROgramming in LOGic« (Programmieren in Logik). Der Grundgedanke bei der Entwicklung von Prolog war die Schaffung einer Sprache, die den Programmierer von allem entlastet, was nicht direkt mit der Logik seines Problems zu tun hat. Doch was ist hier unter »Logik« zu verstehen? Ein Prolog-Programm besteht nicht aus Anweisungen, wie sie von herkömmlichen Programmiersprachen bekannt sind. Statt dessen beinhaltet das Programm eine Beschreibung der Fakten, die über das zu lösende Problem bekannt sind und deren logische Zusammenhänge. Diese Eigenschaft von Prolog bildet im wesentlichen auch den Unterschied zu allen anderen Programmiersprachen: Prolog ist eine deklarative Programmiersprache. Das heißt, der Programmierer gibt nicht den Algorithmus zur Lösung eines Problems an, sondern »lediglich« dessen Beschreibung.

Prolog kennt keine Variablen- und/oder Typvereinbarungen. Da ein Programm in Prolog lediglich aus Regeln (in Prolog werden Fakten auch als Regeln bezeichnet) besteht, kann man sagen, ein solches Programm ist eine große Datenbasis. Betrachten wir dazu das Beispiel aus Bild 2.

Die Fakten (Tatsachen) finden sich im ersten Teil des Beispiels: »vater(..)« und »mutter(..)«. Durch die erste Regel ist festgelegt, daß Dieter der Vater von Helmut ist. Die komplexe Regel »eltern(V,M,K):-vater(V,K),mutter(M,K)« besagt, daß V und M die Eltern von K sind, wenn V der Vater von K und M die Mutter von K ist. Das Komma in der Regel drückt eine Konjunktion (Und-Verknüpfung) aus.

```
/* einfache Regeln (Fakten) : */
vater(Dieter,Helmut) .
vater(Dieter,Edgar) .
vater(Ludwig,Monika) .
vater(Uli,Sabine) .

mutter(Ulla,Helmut) .
mutter(Eva,Edgar) .
mutter(Rosa,Monika) .
mutter(Martha,Sabine) .

/* komplexe Regeln : */
eltern(V,M,K) :- vater(V,K),mutter(M,K) .
```

Bild 2. Ein Beispiel-Programm in Prolog

Programme in Prolog

Dies mag auf den ersten Blick recht verwirrend erscheinen, aber bei näherer Betrachtung erkennt man die Mächtigkeit, die durch diese Art der Programmnotation erreicht wird. Obiges Beispiel stellt tatsächlich ein vollständiges — wenn auch kurzes — Prolog-Programm dar. Es enthält keine einzige Anweisung, Schleife oder explizite Zuweisung wie sie etwa in C oder Pascal auftaucht. Was aber läßt sich mit einem solchen Programm überhaupt anfangen?

Nun — der Anwender kann Fragen an das Programm stellen. Eine solche Frage könnte etwa so aussehen:

```
-? eltern(X,Y,Monika).
```

Prolog geht daraufhin die ihm bekannten Fakten durch und setzt diese in die Regel »eltern« ein. Und zwar in allen möglichen Kombinationen. Die Antwort von Prolog müßte hier lauten:

```
yes
X = Ludwig
Y = Rosa
```

Sie werden sich nun fragen, wie Prolog zu diesem Schluß kommt? Prolog hat quasi einen Schlußfolgerungsmecha-

nismus »eingebaut«. Dieser arbeitet nach dem sogenannten »Backtracking-Prinzip«. Unter Backtracking versteht man die Rückverfolgung eines eingeschlagenen Lösungspfades, der sich unter Umständen als ungeeignet erwiesen hat, bis zu seinem Ursprung und das Suchen nach einer weiteren Alternative. Dies geschieht in unserem Beispiel dadurch, daß Prolog alle Fakten in die Regel einsetzt und solange kombiniert, bis entweder keine Lösungsmöglichkeit mehr besteht, oder eine Lösung gefunden wurde. Selbst in letzterem Fall sucht Prolog weiter, um eventuell weitere Lösungen zu finden. Keine Lösung wird zum Beispiel auf die Frage :

```
-? eltern(X,Y,Hildegard)
```

gefunden, da »Hildegard« nicht in der Wissensbasis unseres Programmes auftaucht.

Anhand dieser Ausführungen läßt sich auch der Unterschied zwischen Prolog und Lisp darstellen. Der Lisp-Programmierer muß sich trotz der schon relativ abstrakten Programmierebene noch Gedanken über den Lösungsalgorithmus machen (was zum Beispiel durch das Schreiben von Funktionen geschieht). Dem Prolog-Programmierer wird diese Arbeit von Prolog abgenommen. Er kann sich voll und ganz den Fakten und den logischen Beziehungen zwischen den Fakten (Regeln) widmen. Prolog eignet sich daher in ganz besonderem Maße für die Programmierung von KI-Anwendungen.

Mit Hilfe von Lisp und Prolog sind bereits heute Programme realisiert, die kurze, leicht verständliche Texte analysieren können. Diese Programme sind dann in der Lage, aufgrund ihrer Wissensbasis und der Analyse der Zusammenhänge in einem solchen Text, kurze Fragen zu diesem Text zu beantworten, ja sogar Schlüsse zu ziehen, die im Text nicht enthalten sind.

Die Programmierer von C und Basic werden jetzt mit Recht behaupten, daß dies auch mit »ihrer« Sprache machbar ist. Nur stellen Lisp und besonders Prolog dazu Mittel zur Verfügung, die Lösungen für derartige Probleme vereinfachen.

(Rainer Arbinger/ Ingolf Krüger/pa)

Wozu Künstliche Intelligenz?

— Erkennung und Übersetzung gesprochener und geschriebener Sprache

— Erkennung und Auswertung von Bildern (Beispiel: Monitorbilder von Überwachungsanlagen)

— Robotik (Steuerung und Überwachung von Robotern und Werkzeugmaschinen)

— Anspruchsvolle und manchmal selbstlernende Spielprogramme (Schach, Dame und andere)

— Beweisführung in Mathematik und Physik

— Expertensysteme

Bild 3. Die Hauptanwendungen der Künstlichen Intelligenz

Literaturhinweis:
Peter Schnupp/Ute Leibrandt, Expertensysteme, Springer Verlag
Thomas Winzer, Künstliche Intelligenz und Robotik, Franzis Verlag
Christine Geißler, Intelligenz für Ihren C 64!, 64er 3/86
Herbert Stoyan, Lisp und Prolog, zwei Programmiersprachen der Künstlichen Intelligenz, 68000er 10/87
Wolfgang Kreitmeir, Prolog, die Programmiersprache der neuen Generation, Pascal 2/87
Karin Krüger-Thielmann, Listenverarbeitung in Prolog, Pascal 3/87
E. Stenzel, Preiswert in die Zukunft, Happy-Computer 12/86
Heimo Ponnath, Nicht nur ein Computerabenteuer, 64er 11/86
Roland Fieger, Prolog 64-Künstliche Intelligenz für den C 64, 64er 11/86



7
Prism – das einzigartige 4096-Farben-Grafikprogramm
Prism, das Malprogramm für den H.A.M.-Modus des Amiga überzeugt durch seine Qualität und Leistungsmerkmale. Sie haben die ganze Bandbreite der 4096 Farbschattierungen des Amiga zur Verfügung – auf einmal und in einem Bild!
Hardware-Anforderung:
Amiga 500, 1000, 2000
mit 512 Kbyte RAM, Kickstart 1.2.
Bestell-Nr. 52570
DM 159,–*
(sFr 142,–*/öS 1990,–*)

8
Zing! – das mächtige CLI-Werkzeug
Mit Zing! haben Sie endlich das gesamte File-System mit Directories und Subdirectories fest im Griff. Die Bedieneroberfläche ist vom Feinsten: Pull-down-Menüs, (Click-)Icons, Funktionstasten und noch viele weitere Optionen.
Hardware-Anforderung:
Amiga 500, 1000, 2000
mit 512 Kbyte RAM, Kickstart 1.2.
Bestell-Nr. 52571
DM 189,–*
(sFr 169,–*/öS 2290,–*)

9
Zing Keys! – Ihr ganz persönlicher Amiga
Zing Keys! bietet jetzt ein formbares Software-Werkzeug mit erstaunlichen Fähigkeiten. Mit Zing Keys! machen Sie aus Ihrem Amiga das variable System, das Sie sich schon immer wünschen. Alle Tasten sind nach Wunsch belegbar, und die Belegung ist natürlich jederzeit abspeicherbar. Durch die Belegung von Hot-Keys haben Sie mit Zing Keys! ein Multitaskingsystem par excellence!
Hardware-Anforderung:
Amiga 500, 1000, 2000
mit 512 Kbyte RAM, Kickstart 1.2.
Bestell-Nr. 52572
DM 109,–*
(sFr 98,–*/öS 1390,–*)

10
CLImate 1.2
Mit CLImate 1.2 können Sie endlich die Befehle des Command-Line-Interface benutzerfreundlich per Mausklick verwenden! Eine sehr übersichtliche Bildschirmdarstellung, die Bedienung aller Befehle mit der Maus und die Unterstützung von drei externen Laufwerken (3½" oder 5¼"), Festplatten, RAM-Disk machen das Programm zu einem unentbehrlichen Werkzeug.
Hardware-Anforderung:
Amiga 500, 1000 oder 2000 mit mindestens 512 Kbyte Hauptspeicher. Empfohlene Hardware: Farbmonitor. Software-Anforderung: Kickstart 1.2 (oder ROM bei Amiga 500 und 2000), Workbench 1.2.
Bestell-Nr. 51653
DM 79,–*
(sFr 72,–*/öS 990,–*)

11
Superbase Amiga
Superbase Amiga vereint eine neuartige, äußerst benutzerfreundliche Bedienung mit Pull-down-Menüs, Fenstern und Maussteuerung mit der enormen Leistungsfähigkeit einer relationalen Dateiverwaltung: Sie können Datenbanken einfach und schnell aufbauen, Daten übersichtlich verwalten, z.B. als Tabelle oder Formular, beliebige Datenkategorien auswählen, Dateien verknüpfen und einbinden, Bilder und Grafiken darstellen, einzigartige Dia-Shows erstellen oder Grafik-Datenbanken verwalten.
Bestell-Nr. 51636
DM 249,–*
(sFr 199,–*/öS 2990,–*)

12
VideoScape 3D – ein neues Programm für die Amiga
Mit VideoScape 3D können Sie dreidimensionale Objekte aus verschiedenen Blickwinkeln ansehen und durch Hinzufügen von Kamerafahrten und frei wählbarem Lichteinfall einen realistischen Computer-Videofilm erstellen. Das Programm ist an die PAL-Auflösung angepaßt und wird mit deutschem Handbuch ausgeliefert.
Bestell-Nr. 51671
DM 385,–*
(sFr 345,–*)

13
DevPack Assembler
Ein Entwicklungspaket mit integriertem Editor/Assembler, symbolischem Debugger und schnellem Linker zum Einbinden von Hochsprachen-Modulen. Erzeugt direkt ausführbare Programme!
Bestell-Nr. 51656
DM 148,–*
(sFr 134,–*)

* Unverbindliche Preisempfehlung.

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656
ÖSTERREICH: Rudolf Lechner & Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526
Ueberreuter Media Verlagsges. mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.

Konkurrenz für Amiga-Basic

Ein Basic-Interpreter muß schon einiges bieten, um sich gegen das kostenlose Amiga-Basic durchzusetzen. SAM-Basic und True-Basic sind angetreten. Haben die beiden Interpreter eine Chance?

Das mit dem Amiga gelieferte Basic von Microsoft ist nicht mehr das einzige Basic für Ihren Computer. Inzwischen gibt es diese Sprache von zwei weiteren Herstellern. Wir haben True-Basic und SAM-Basic für Sie getestet. Es gilt die Frage zu beantworten, ob der Leistungsumfang beider Interpreter einen Basic-Programmierer dazu bewegen kann, sich ein neues Basic anzuschaffen, obwohl der Amiga zusammen mit einem leistungsstarken Interpreter ausgeliefert wird?

Ein ernstzunehmender Konkurrent für das Amiga-Basic muß schon Besonderes bieten, denn niemand wird Geld für etwas ausgeben, was er bereits hat. Amiga-Basic ist also unser Maßstab, an dem sich alle anderen Interpreter messen lassen müssen.

Die beiden uns vorliegenden Basic-Versionen verfolgen in der Erweiterung des Leistungsumfangs gegenüber Amiga-Basic zwei unterschiedliche Konzeptionen.

Komfortabel und kompatibel

Da wäre zunächst das True-Basic, dessen Entwicklungsschwerpunkt neben einer Erweiterung des vom Amiga-Basic bekannten Sprachumfangs auf der Kompatibilität zu anderen Computern liegt. True-Basic gibt es nicht nur für den Amiga, sondern unter anderem auch für MS-DOS-Computer. Ein auf dem Amiga entwickeltes True-Basic-Programm läuft also im Normalfall auch auf einem IBM-PC.

Die Philosophie hinter SAM-Basic ist eine vollkommen andere. David Simons, der den C64-Fans noch als Programmierer des Simons-Basic bekannt sein dürfte, hat vor allem Wert darauf gelegt, die Fähigkeiten der Amiga-Hardware auszunutzen. Diese unterschiedlichen Zielsetzungen gilt es bei der nun folgenden Betrachtung des Leistungsumfangs beider Sprachen im Auge zu behalten.

Wenden wir uns zunächst dem »wahren« Basic zu. Eine positive Überraschung erlebt man bei der näheren Untersuchung der mitgelieferten Dokumentation. Sie besteht aus zwei, jeweils etwa 300 Seiten dicken Handbüchern, die in leicht verständlichem Englisch geschrieben sind. Der User-Guide dient als Einführung für Basic-Neulinge, während das Reference-Manual als Nachschlagewerk gedacht ist. Beide Bücher sind didaktisch sehr gut aufbereitet und lassen eigentlich keine Wünsche offen, die man an ein gutes Handbuch stellt. Oder um es anders auszudrücken: Die zu True-Basic mitgelieferte Dokumentation ist so, wie man sie sich bei Amiga-Basic gewünscht hätte.

Doch nun zum Programm selbst. True-Basic wird auf einer erfreulicherweise nicht kopiergeschützten Diskette geliefert. Der Programmstart erfolgt durch die bereits von Amiga-Basic bekannten Methoden. Der Editor hält sich in seiner äußeren Aufmachung ebenfalls an das Vorbild Amiga-Basic. Er wurde jedoch um einige sinnvolle Befehle, wie etwa »Suchen und Ersetzen« erweitert. Ebenso bereitet das Einlesen von Programmen in das zur Zeit editierte Programm durch den »Include«-Befehl keine Schwierigkeiten mehr. Auch wenn der Editor noch nicht den Befehlsumfang einer Textverarbeitung bietet, wird durch die Aufwärtskompatibilität zum Editor des Amiga-Basic und vor allem durch die Such- und Ersetzungsbefehle eine komfortable Programmerstellung gewährleistet.

Ein mächtiger Befehl des True-Programmeditors ist die DO-Anweisung. Sie gestattet es, das aktuelle Programm zu bearbeiten.

Strukturiertes Basic

So ruft zum Beispiel der Menüpunkt DO FORMAT das Programm FORMAT auf. Dieses Programm benutzt das Programm im Editor als Eingabe und wandelt alle darin vorkommenden Basic-Befehle in Großschreibung um. Der Anwender kann sich allerdings auch selbst Programme schreiben, die dann mit DO aufgerufen werden. Ein solches Programm könnte etwa ein Präprozessor sein, wie man ihn bei der Programmierung in der Sprache C kennt.

Gegenüber Amiga-Basic haben die Entwickler den Befehlsumfang des Interpreters stark erweitert. Wie auch bei seinem Gegenstück sind bei True-Basic Zeilennummern aufgrund der mächtigen strukturierten Befehle überflüssig. Mit FOR..NEXT-, DO..LOOP-, DO..UNTIL- und DO..WHILE-Schleifen stehen dem Anwender leistungsfähige Anweisungen zur Verfügung. Auch an einen Befehl zum vorzeitigen Verlassen einer Schleife wurde gedacht. Zusätzlich zu den bekannten IF..THEN..ELSEIF-Strukturen wird auch noch eine CASE-Anweisung angeboten, so daß in bezug auf strukturierte Befehle keine Wünsche mehr offen bleiben.

Die Grafikanweisungen lassen die Herzen engagierter Grafikfans höher schlagen. Eine enorme Erleichterung bei der Programmierung ist die freie Wahl des Koordinatensystems der Darstellungsfläche. True-Basic arbeitet mit Benutzerkoordinaten und nicht mit den üblichen Bildschirmkoordinaten. Der Programmierer kann so die Anzahl der Punkte auf der X- beziehungsweise Y-Achse (fast) frei bestimmen. Hinzu kommt, daß viele Farben bereits mit Namen vorbesetzt sind. So schaltet zum Beispiel »SET COLOR "red"« die aktuelle Zeichenfarbe auf rot. Selbstverständlich ist man bei den Ausgaben, die ein Programm liefert, nicht nur auf ein Window beschränkt.

Analog zu den grafischen Befehlen GET und PUT in Amiga-Basic gibt es im True-Basic die Befehle BOX SHOW, BOX CLEAR und BOX KEEP. Dabei ist der BOX KEEP-Befehl, der ein grafisches Objekt auf dem Bildschirm an eine Variable zuweist, wesentlich einfacher zu handhaben, als die entsprechende Anweisung des Amiga-Basic, bei dem man die Größe des Variablenfeldes erst noch selbst berechnen muß. Eine weitere schöne Eigenschaft von True-Basic sind die sogenannten PICTURES. Dies sind Unterprogramme, die Bilder zeichnen und beliebig auf dem Bildschirm positionieren.

Bei all den erwähnten Pluspunkten darf allerdings nicht unerwähnt bleiben, daß die Übertragbarkeit von True-Basic-Programmen auf andere Rechner auch ihren Tribut fordert. Befehle zum Ansprechen von Sprites und Bobs sucht man im Grundpaket vergebens. Ebenso sind die Möglichkeiten der Tonerzeugung noch spärlicher als in Amiga-

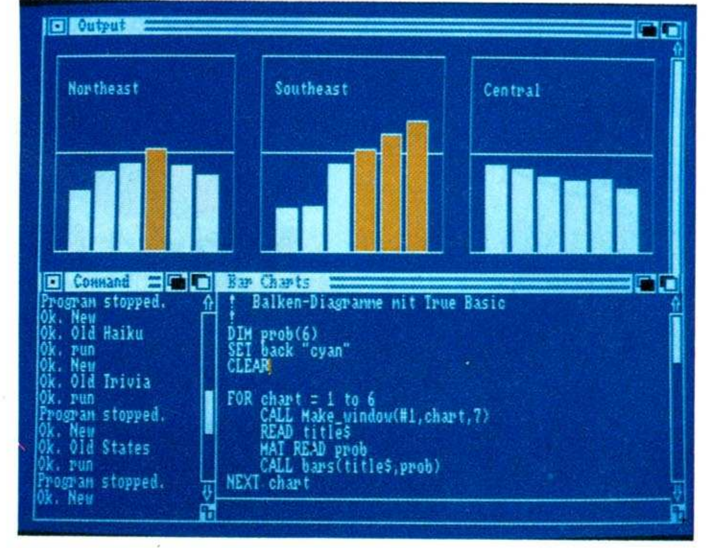


Bild 1. Die drei Fenster des True-Basic-Interpreters

Basic. Für die erfahreneren Programmierer gibt es allerdings Möglichkeiten, dieses Manko zu überwinden. Darauf gehen wir später noch ein.

Zunächst werfen wir noch einen Blick auf die Rechenbefehle von True-Basic. Die große Anzahl der zur Verfügung stehenden Funktionen (selbst so seltene Funktionen wie die Poisson-Verteilung sind vorhanden) dürfte kaum noch Wünsche offenlassen. Nicht nur dem Programmierer in Sachen Grafik werden die Manipulationsmöglichkeiten zweidimensionaler Arrays (Matrizen) gefallen. So können Matrizen addiert, multipliziert, invertiert und transponiert werden. An eine Funktion zur Berechnung der Determinante wurde ebenfalls gedacht.

Die vorhandenen Stringbefehle sind ebenfalls umfangreicher und komfortabler als bei Amiga-Basic. Die entsprechenden Funktionen zur Bearbeitung von Menüs mit Hilfe der Maus sind ebenso vorhanden. Allerdings haben die Menüs (wohl wieder aus Gründen der Übertragbarkeit) nicht das vom Amiga gewohnte Bild der Pull-Down-Menüs.

Als weiterer Vorteil gegenüber Amiga-Basic muß gelten, daß eigendefinierte Funktionen nicht mehr auf eine Zeile beschränkt sind. Befinden sich Funktionen oder Unterprogramme hinter dem END-Statement, so sind ihre Variablen lokal. Derartige Unterprogramme können sogar rekursiv aufgerufen oder zu Libraries zusammengefaßt werden. Doch damit nicht genug: Die in den Libraries enthaltenen Programme müssen nicht unbedingt in Basic geschrieben sein — auch C- oder Assemblerprogramme sind möglich.

Ein Interpreter für den Amiga

Die nötigen Include-Files mit den vordefinierten Headern werden mitgeliefert. Somit ist es ohne weiteres möglich, sich die fehlenden Sprite- und Musikbefehle selbst zu schreiben. Etwas Erfahrung in Systemprogrammierung sollten Sie dann allerdings schon besitzen.

Bevor wir uns jetzt mit SAM-Basic auseinandersetzten, noch einige Bemerkungen zu den Variablentypen. Erstaunlicherweise kennt True-Basic nur deren zwei: Zahlen und Strings. Dabei können Strings bis zu 1048575 Zeichen enthalten, aus Gründen der Übertragbarkeit auf andere Rechner sollte man sich aber auf 32000 Zeichen beschränken. Der verwendbare Zahlenbereich liegt bei übertragbaren Programmen zwischen —1E-99 und +1E99, bei Beschränkung auf den Amiga bei —1E-309 und +1E308. Die dabei erreichte Rechengenauigkeit beträgt mindestens 10 Stellen. Dies ist auch der Grund dafür, weshalb True-Basic im Benchmark 2 schlechter abschneidet. Falls Sie sich wundern sollten, weshalb es keine Integer-Variablen gibt: True-Basic ist kein Interpreter, sondern ein Compiler. Er erzeugt allerdings keinen echten Maschinencode, sondern einen sogenannten Zwischencode. Während des Compilierens stellt True-Basic fest, welche Variablen als Ganzzahlvariablen benutzt werden können und nimmt dann die entsprechenden Zuweisungen vor. Dies macht sich zum Beispiel im Bench-

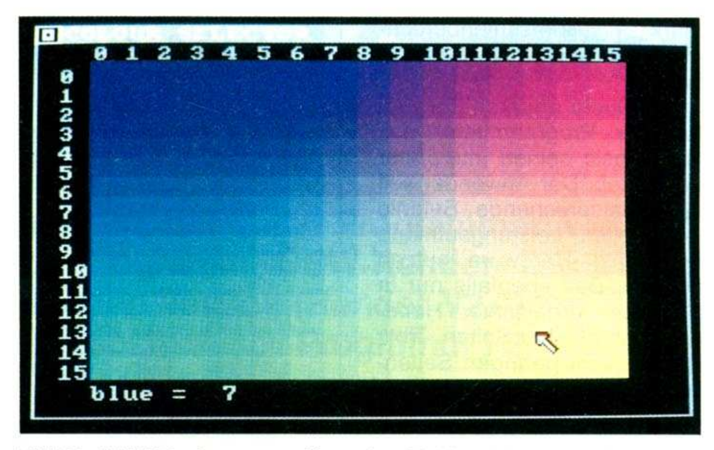


Bild 2. SAM-Basic unterstützt den Hold & Modify-Modus

mark 1 (Primzahlenberechnung) ganz deutlich bemerkbar.

Mit einem anderen Konzept hat der Programmierer des SAM-Basic seinen Interpreter entwickelt. Der Anwender soll über entsprechende Befehle den vollen Leistungsumfang der Amiga-Hardware nutzen können. Das ebenfalls nur in Englisch erhältliche Handbuch umfaßt 180 Seiten. Trotz dieses nicht geringen Seitenumfanges sollte man aber eher von einem Stichwörterverzeichnis sprechen, denn die Erklärungen zu den einzelnen Befehlen sind ziemlich knapp gehalten. Daraus können Sie bereits den Schluß ziehen, daß es sich bei SAM-Basic um eine mächtige Befehlserweiterung handelt, die den Bedürfnissen erfahrener Basic-Programmierer entgegenkommt. Da wir schon dabei sind: SAM-Basic stellt nicht nur an den Kenntnisstand des Programmierers ziemliche Ansprüche, sondern auch an den Speicher. Um alle Möglichkeiten wirklich voll (das heißt gleichzeitig) ausschöpfen zu können, wird ein Speicherausbau von mindestens 1 MByte benötigt.

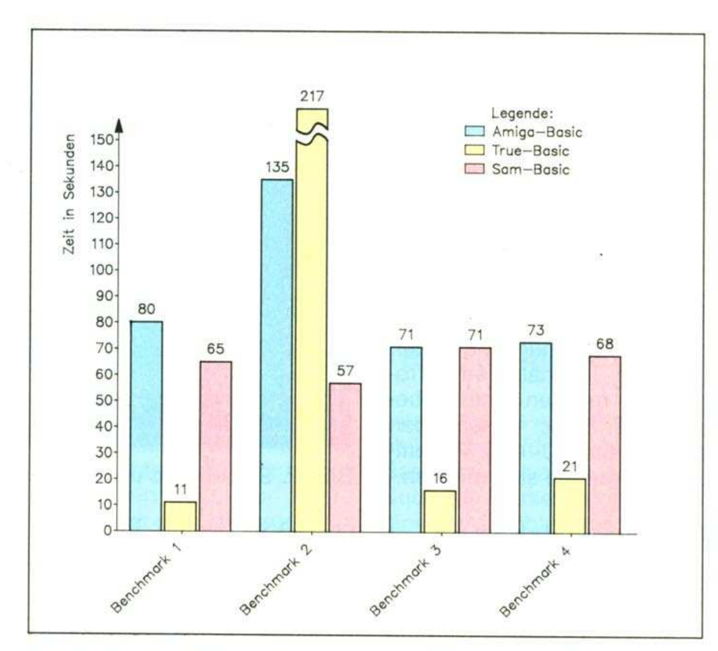


Bild 3. Die drei Basic-Interpreter im Benchmark-Test

Spartanischer Editor

Der in SAM-Basic eingebaute Editor ist leider sehr spartanisch ausgefallen. Dieser Mangel ist auch dem Entwickler bewußt, denn er hat mit der Anweisung EDITOR IS einen Befehl implementiert, mit deren Hilfe sich andere Editoren in das System einbinden lassen.

Strukturierte Befehle stehen praktisch im gleichen Umfang wie bei True-Basic zur Verfügung. Daher ist es absolut unverständlich, daß bei SAM-Basic die Verwendung von Zeilennummern unbedingt erforderlich ist. Zwar gibt es auch hier die Befehle GOTO und GOSUB, doch ist es — ähnlich wie in Amiga-Basic — möglich, Labels zu definieren. Selbst wenn SAM-Basic die Zeilennummern zu seiner internen Verarbeitung benötigen sollte, so ist es zumindest schlechter Stil, diese Verwaltungsarbeit dem Benutzer aufzubürden. Überhaupt hinterläßt SAM-Basic einen etwas zwiespältigen Eindruck. Die Programmierer hatten sehr viele und gute Ideen, bei der Implementierung aber scheinbar nicht den Mut, diese auch vollständig in die Tat umzusetzen. Es ist sehr deutlich zu merken, daß SAM-Basic noch den (Un-) Sitten alter Basic-Versionen nachhängt. Neben den Zeilennummern macht sich dies auch bei der eher zurückhaltenden Unterstützung der Maus bemerkbar. Bedauerlich ist auch, daß das von den Entwicklern für SAM-Basic gewählte Bibliotheksformat nicht mit den Amiga-Libraries kompatibel ist und ein entsprechender Befehl zum Einbinden der Amiga-Libraries fehlt. War dies bei True-Basic aus Kompatibilitätsgründen noch zu verzeihen, ist es bei SAM-Basic unentschuldbar.

SAM-Basic ist ein echtes Multitasking-System, bei dem bis zu 32 Basic-Programme gleichzeitig ablaufen können. Ähnlich wie auch True-Basic kann SAM-Basic erweitert werden. Die Erweiterungen müssen sich jedoch an das vorgegebene Library-Format halten. Daher ist es nicht möglich, diese Erweiterungen in Basic selbst zu schreiben. Auch wenn die String- und Numericfunktionen von SAM-Basic nicht so umfangreich wie in True-Basic sind, so können sie sich durchaus mit denen von Amiga-Basic messen.

Die beiden wichtigsten Einschränkungen von True-Basic treten nicht auf: Sowohl die Unterbrechungsverarbeitung als auch die Fähigkeit, Sprites und Bobs zu verwalten, sind vorhanden. Vor allem Sprites und Bobs lassen sich mit einem Komfort verarbeiten, der Amiga-Basic-Programmierer wohl vor Neid erblassen läßt.

Die Unterbrechungsverarbeitung gestattet die komfortable Abfrage von Maus und Menüs. Aber auch die Abfrage der Tastatur ist ohne weiteres möglich. Die normalen Grafikbefehle entsprechen im wesentlichen dem von Amiga-Basic gesetzten Standard. SAM-Basic hat jedoch noch ein besonderes Bonbon zu bieten: Die Entwickler haben zusätzlich Befehle zum Erstellen und Bearbeiten von 3D-Grafiken implementiert. Diese Befehle ersparen viel Handarbeit, die bei entsprechenden Programmen in Amiga-Basic oder True-Basic auszuführen wäre.

Natürlich verwaltet SAM-Basic Windows und die unterschiedlichen Screen-Formate (in eingeschränkter Form auch den HAM-Modus). Entsprechende Funktionen erleichtern den Umgang mit IFF-Dateien.

So stehen Befehle zum Laden und Speichern von Bildern im IFF-Format zur Verfügung. Im Gegensatz zu den beiden anderen Interpretern ist der Wechsel von Schriftarten ebenfalls vorgesehen. Zu guter Letzt sei noch erwähnt, daß in der inzwischen vorliegenden Version 3.1 von SAM-Basic zusäzliche Befehle eingebaut wurden, die es ermöglichen, den Copper direkt zu manipulieren. Für Kenner der Amiga-Hardware ergeben sich damit auch von Basic aus Möglichkeiten, die früher nur von Sprachen wie etwa C zu realisieren waren. Eine scheinbar typische Krankheit von Basic-Systemen auf dem Amiga darf allerdings nicht unerwähnt bleiben: Auch bei SAM-Basic ist die Unterstützung der Sound-Eigenschaften nur in geringem Maße vorhanden.

Bis jetzt wurde eigentlich nur der Sprachumfang der unterschiedlichen Basic-Versionen besprochen. Doch der sollte nicht das einzige Kriterium zur Beurteilung einer Sprache liefern. Von mindestens ebenso großer Bedeutung ist die Ausführungsgeschwindigkeit von Programmen. Daher wurden einige, möglichst gleichlautende Programme auf den unterschiedlichen Basic-Versionen auf Ihre Geschwindigkeit hin untersucht. Solche Tests werden auch als Benchmark bezeichnet. Dabei ist Benchmark 1 das bekannte Sieb des Eratosthenes zur Primzahlenbestimmung. In unserem Fall wurden die Primzahlen kleiner als 10000 bestimmt (nicht jedoch ausgegeben). Dieser Test ist vor allem geeignet, die Ausführungsgeschwindigkeit von IF-Abfragen, Schleifen und Integerarithmetik zu ermitteln. Der nächste Benchmark testet die Ausführungsgeschwindigkeit von Gleitkommaoperationen. Dazu wurde in einer Schleife die Funktion sin(exp(log(x))) berechnet. Da die Berechnung dieser verschachtelten Funktion den Hauptanteil der Rechenzeit einnimmt, ist hier kein Vorteil eines Compilers über einen Interpreter zu erwarten. Ganz im Gegenteil macht sich hier die größere Rechengenauigkeit von True-Basic bemerkbar. Der dritte Benchmark testet nun nicht die komplexeren numerischen Funktionen, sondern die einfache Multiplikation und Division von Gleitpunktzahlen. Der vierte und letzte Benchmark testet die Verarbeitung von Strings, indem ein vorgegebenes Stringfeld sortiert wird. Nicht getestet wurden die besonderen Amiga-Eigenschaften, da anzunehmen ist, daß hier alle drei Basic-Versionen die eingebauten Amiga-Funktionen benutzen.

Selbstverständlich konnten wir aus Platzgründen nicht auf alle Eigenschaften der Basic-Versionen eingehen. Dennoch läßt sich aus den bisherigen Aufführungen unsere Wertung nachvollziehen: Sowohl True-Basic als auch SAM-Basic übertreffen in ihren Fähigkeiten Amiga-Basic. Dabei haben die jeweiligen Entwickler aber unterschiedliche Akzente gesetzt: Bei True-Basic wird Wert auf Kompatibilität gelegt, während SAM-Basic mehr auf die speziellen Amiga-Eigenschaften eingeht.

Fortsetzung auf Seite 132

Tolle Bilder und Grafiken fast geschenkt

Public Domain-Software hilft, bei guten Programmen viel Geld zu sparen. Das Freesoft-Programm »DBW-Render« liegt nun in einer erheblich verbesserten Version (Render V2.0) vor. Lesen Sie, wie gut dieses »fast kostenlose« Programm ist.

Gerade auf dem Amiga mit seiner hohen Grafikauflösung und der Vielzahl von darstellbaren Farben ist die Anwendung von Grafikprogrammen nahezu ein Muß. Ein Programm, das diese Möglichkeiten ausnutzt und sogar mehr als der Amiga kann, ist das Ray-Tracing-Programm »DBW-Render«, das nun in der verbesserten Version 2.0 vorliegt. Mit Hilfe von DBW-Render ist es ein leichtes, ähnlich wie mit »Sculpt 3D« oder »Silver« natürlich wirkende Grafiken, auch als Ray-Tracer bezeichnet, zu berechnen (Bild 1).

Dabei sind die Mittel und Möglichkeiten, die einem Render zur Verfügung stellt, geradezu phänomenal: maximale Bildgröße 1024 x 1024 Pixel; beliebige Farbanzahl (maximal 16 Millionen Farben); zwei- und dreidimensionaler Text kann mit Strukturen versehen und direkt in die Grafik eingesetzt werden (Bild 1); makro- und postscriptfähig; komfortable Eingabe.

Dies bedeutet, daß der Käufer des Programms auch für zusätzliche Grafikkarten gerüstet ist (eine Grafikkarte mit 1024 x 1024 Pixel und zwei Millionen Farben soll von Mimetics angekündigt sein).

Doch wie kann ein Ray-Tracer erstellt werden? Sicher werden Sie denken, daß es kompliziert ist, ein so echt wirkendes Gebilde wie etwa den Körper aus Bild 2 zu konstruieren. Doch weit gefehlt: Es ist nicht viel Vorbereitungszeit nötig. Render stellt eine Programmiersprache zur Verfügung, mit der der Anwender die benötigten Parameter in eine Textdatei schreibt, die später compiliert und ausgeführt wird. Das Bild wird also nicht gezeichnet, sondern programmiert. Dies klingt zwar umständlich, ist aber in Wirklichkeit relativ einfach. So sieht die Angabe der Größe eines gedachten Bildes etwa so aus:

```
SCAN FROM 0 to 400 ROWS
400 COLUMNS 320
```

Dies bedeutet, daß insgesamt 400 Zeilen existieren, die auch alle 400 berechnet werden. Das Bild wird 320 Pixel breit. Ebenso leicht läßt sich bestimmen, von wo aus der Betrachter auf den Körper blickt.

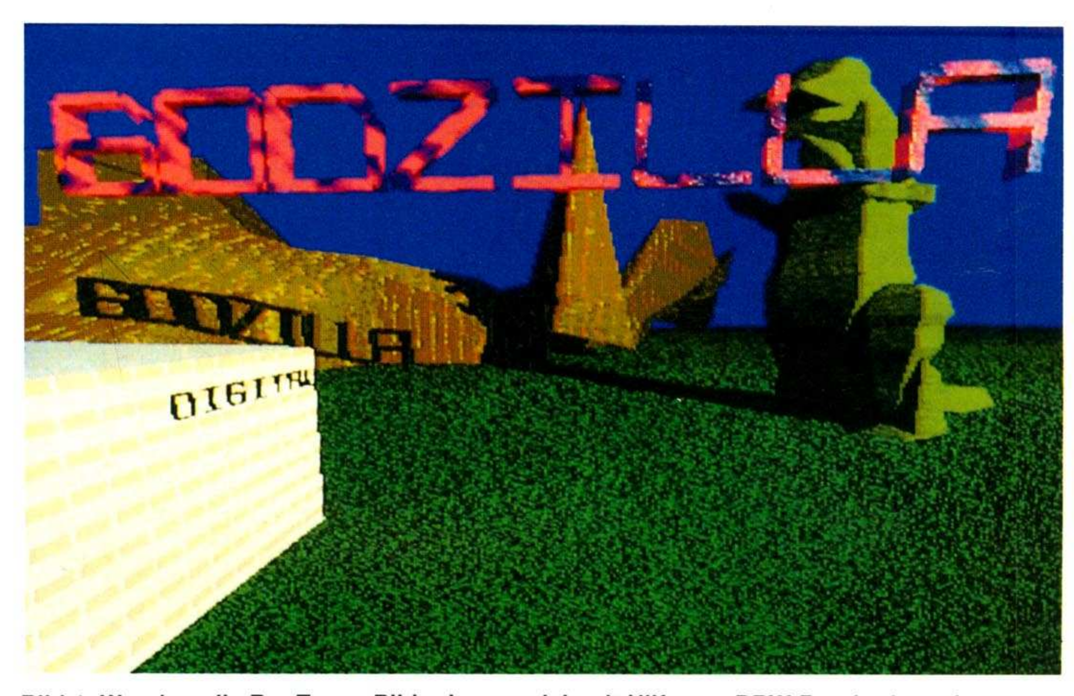


Bild 1. Wundervolle Ray-Tracer-Bilder lassen sich mit Hilfe von DBW-Render berechnen

Auch die einzelnen Körper sind mit wenigen Eingaben fixiert. Ein Kreisring erfordert nur drei Textzeilen:

```
RING Brick
RADIUS 0 64
RADIUS 1 128
```

Die zweite und dritte Zeile geben den Innen- und Außendurchmesser des Kreisrings an. Die erste Zeile bestimmt, welche Oberflächenstruktur das Material besitzt, in diesem Fall eine Ziegelsteinmauer. Das Programm stellt eine Vielzahl von Strukturen zur Verfügung: Neben der Ziegelstruktur etwa noch Glas, Spiegel, Metall, Gras oder Holz. Sogar ein IFF-Bild läßt sich einbinden, das später um den Körper herum gelegt wird.

Selbstverständlich kann der Benutzer auch Makros definieren, die zu einem Gesamtbild zusammengefügt werden können. Mit den passenden Befehlen, die man in einem fast 40seitigen, leider englischsprachigen Manual findet, ist das kein Problem. Auf der Diskette sind einige Demo-Dateien enthalten, mit denen herumexperimentiert werden kann, um einige Grundprinzipien der Programmierung von Render-Grafiken kennenzulernen. Mit wenigen Programmzeilen können so innerhalb kurzer Zeit die ersten Grafiken berechnet werden, die danach im IFF-Format vorliegen.

Das umfangreiche Programmpaket, das zu einem Preis von 34 Mark (per Nachnahme 38 Mark) bei untenstehender Adresse zu erhalten ist, umfaßt insgesamt fünf Disketten: Die Version 1.0, die neue Version 2.0, eine Demodiskette sowie zwei weitere Disketten, die eine komplette Liste der meisten PD-Serien enthalten. Außerdem liegt eine Kopie der Render-V2.0-Anleitung bei. Alles in allem erhält man für einen fairen Preis ein höchst leistungsfähiges Grafikpaket, mit dem sich ohne großen Aufwand so schöne Grafiken, wie auf den Bildern zu sehen sind, konstruieren lassen. (dm)

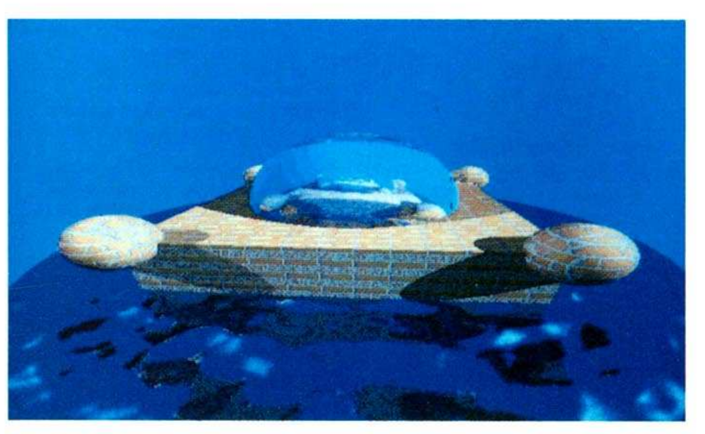


Bild 2. Dieser plastisch wirkende Körper benötigt nur wenige Textzeilen zur Erstellung der Anweisungsdatei

Stefan Ossowski, Veronikastr. 33, 4300 Essen 1, Tel. 0201/788778, Preis: 34 bzw. 38 Mark

Die Public Domain-Seite

Immer mehr gute Programme werden von Monat zu Monat als Public Domain angeboten. Auch diesen Monat gibt es wieder einige Disketten, die wir Ihnen für Ihre Freesoft-Sammlung empfehlen können.

Auf Public Domain, auch Freesoft genannt, finden sich für wenig Geld häufig viele und gute Programme, die teilweise unentbehrliche Tools und Utilities darstellen. Auf dieser Seite stellen wir Disketten vor, die Sinnvolles und Nützliches enthalten und ohne Bedenken gekauft werden können.

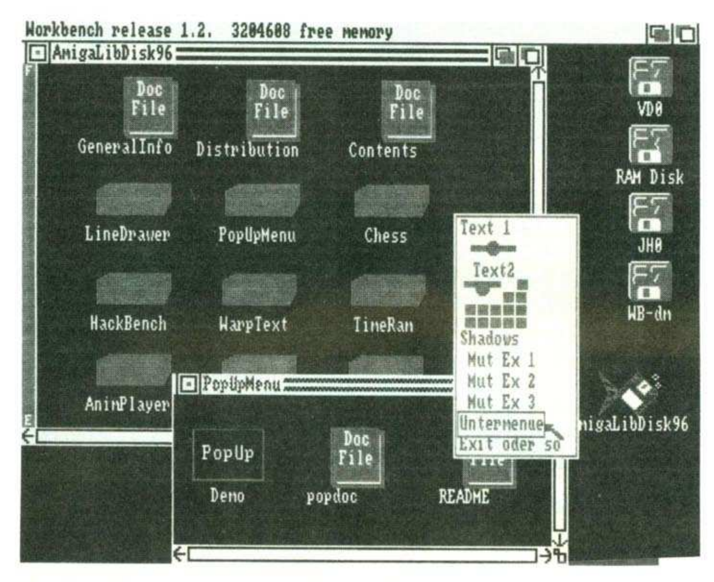
Wer sich gerne Bilder ansieht, dem wird sicher die »RW 8« der neuen PD-Serie von Rainer Wolf gefallen. Auf dieser Diskette befindet sich eine Diashow mit schön digitalisierten H.A.M.-Bildern, bei deren Ablauf nebenbei eine gut digitalisierte Musik läuft. Für Grafikfreunde ist diese Diskette sicher ihrer guten Bilder wegen ein Muß.

Spielefreunde haben sicher ihre Freude an der »RW 2«. Auf dieser Diskette finden sich neben den bereits von anderen PD-Disketten her bekannten Spielen »Cluedo« und »Cycles« zwei neue Spiele: Zum einen »Kampf um Eriador«, ein Strategiespiel mit einer guten Grafik, und »Börse«, das altbekannte Spekulationsspiel. Sicher nimmt diese Diskette aber keinen Platz in der PD-Sammlung weg, da ab und zu bestimmt mal gespielt wird.

Eine andere Serie ist die der »Tampa Bay-Amiga-Group« (TBAG). Auf der ersten Diskette, der »TBAG 1«, befinden sich drei IFF-Bilder, verschiedene Demoprogramme wie etwa »Ball«, das dem Mauspointer ein anderes Aussehen gibt, oder »150Windows«, das , wie der Name schon sagt, 150 Fenster öffnet und danach wieder schließt. Aber auch verschiedene Spielprogramme erhält man mit dem Kauf dieser Diskette: Ein Buchstabenpuzzle, das bekannte PD-Spiel »CosmoRoids« sowie »Crystal Vision«. Die Diskette ist zwar nicht lebenswichtig, doch sind die Programme sicher den Geldbetrag wert.

Sowohl bekannte wie auch völlig neue Grafiken enthält die »Tornados 4«. Mehr als ein Dutzend gelungener Bilder werden in einer Diashow angezeigt. Für alle, deren IFF-Bilderdisketten noch nicht gänzlich gefüllt sind, ist diese Diskette sicher ein gefundenes Fressen. Wir meinen: Für Grafikfreaks gerade richtig.

Möchten Sie gerne in Assembler programmieren, verfügen aber über keinen geeigneten Assembler? Auf »Tornados 6« befindet sich ein kompletter, wenn auch einfacher Assembler nebst Zusatzlibraries und Tools. Sicher kann der Assembler nicht mit Programmen wie beispielsweise dem Seka-Assembler mithalten, jedoch genügt er, um in Assembler zu programmieren. Unsere Meinung: Diese Diskette sollte jeder besitzen, der in Assembler programmiert oder programmieren möchte.



Unglaublich, aber wahr. »PopUpMenu« erlaubt, daß an der aktuellen Mausposition ein Menü erscheint

»Tornados 9« ist eine nützliche Diskette, die eine Vielzahl von verschiedenen Programmen enthält: Etwa eine kleine Digitaluhr, die wenig Platz auf dem Bildschirm einnimmt oder »SetFont«, das es gestattet, mit verschiedenen Zeichensätzen im CLI-Fenster zu arbeiten. Das Programm »Journal« erlaubt es, verschiedene Tasten- und Mausbewegungen in einer Datei aufzuzeichnen, die später wieder abgespielt werden können. Dies dient beispielsweise dazu, Demoprogramme zu erzeugen. Oder möchten Sie Ihren Computer gegen fremde Benutzung schützen? Kein Problem mit »KeyLock«. Erst durch Eingabe eines Paßworts kann die Maus oder Tastatur wieder benutzt werden. Auch »PALTest« zum Testen der Version (PAL oder NTSC) des Computers beziehungsweise der Workbench soll erwähnt werden. Alles in allem ist die »Tornado 9« sicher eine Freesoft-Diskette, die ihren Platz in jeder PD-Sammlung finden kann.

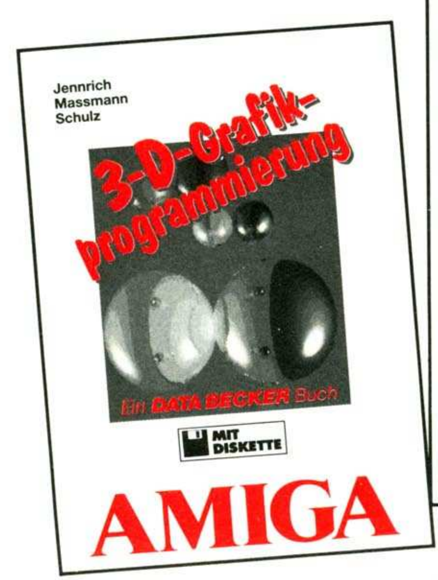
Viele einzelne Dateien erhält der Käufer der Fish »67«. Beispielsweise »AmCat«, ein Programm, mit dem sich Ihre Disketten katalogisieren lassen. Oder das durchaus leistungsstarke Terminalprogramm »Comm«, das über genügend Möglichkeiten verfügt, um sinnvoll kommunizieren zu können. Programmierer interessieren sich sicher für den Taschenrechner »HexCalc«, der durch seine verschiedenen Zahlensysteme eine Unterstützung darstellt. »DUX5« ist ein DOS-Hilfsprogramm, welches ähnlich CLI-Mate erlaubt, ohne großen Aufwand Diskettenoperationen wie Kopieren, Löschen oder Formatieren zu erledigen. Das interessante Grafik-Demo »Bouncer« rundet genauso wie das Echtzeit-Grafik-Demo »RTCubes« oder die kleine Adreßverwaltung »PersMait« diese sicher gute PD-Disk ab.

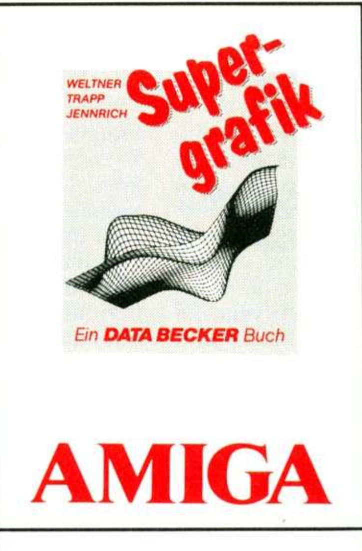
Sehr interessant für C-Programmierer ist auch Fish »96«. Dort findet sich das Programm »PopUpMenu«, welches erlaubt, daß an der aktuellen Position des Mauspointers ein Menü erscheint (Bild). Selbstverständlich ist auch der C-Quellcode dazu enthalten. Ein weiteres Programm erzeugt einen H.A.M.-Screen, der als Testbildschirm verwendet werden könnte. Interessant ist vor allem »TimeRAM«: Es dient zum Messen der Geschwindigkeiten des Fast- und Chip-RAM. So läßt sich erkennen, ob das Fast-RAM wirklich schneller ist und wenn ja, um wieviel.

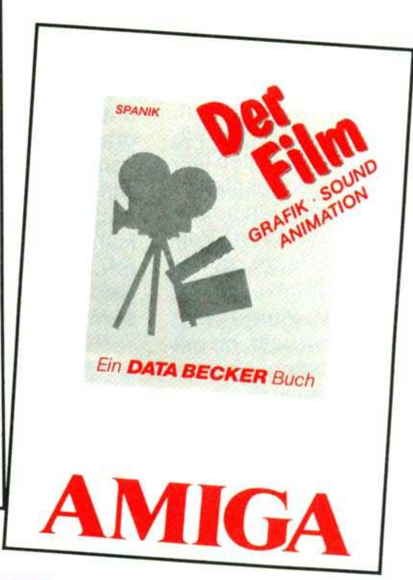
Midi-Programmierer werden an Fish »101« ihre Freude haben: Die Diskette enthält eine Library, mit der verschiedene Midi-Anwendungen programmiert werden können. Ebenfalls interessant ist der »Icon-Assembler«, der es gestattet, zwei verschiedene Icons zu verknüpfen oder bestehende Icon-Verbindungen zu ändern. Der englische Rechtschreibprüfer »MicroSpell« und verschiedene Demoprogramme runden das Bild dieser Diskette ab. Unsere Meinung: Der »IconAssembler« ist allein schon den Kauf der Fish »101« wert. (dm)

Anbieter von Public Domain-Disketten (ohne Anspruch auf Vollständigkeit):
Stefan Ossowski, Veronikastr. 33, 4300 Essen, Tel. 0201/788778
Atlantis, Dunantstr. 53, 5030 Hürth, Tel. 02233/31066
Datentechnik Bittendorf, Postfach 100248, 6360 Friedberg, Tel. 06031/61950
Fastworks, Fichtestr. 16, 5090 Leverkusen 1, Tel. 0214/92802
EcoSoft AG, Kaiserstr. 21, 7890 Waldshut, Tel. 07751/7920
Christian Bellingrath, Trift 10, 5860 Iserlohn, Tel. 02371/24192
Rainer Wolf Soft- und Hardwareversand, Deipe Stegge 187, 4420 Coesfeld, Tel. 02541/2874
Uwe Schmielewski, Haroldstr. 71, 4100 Duisburg 1, Tel. 0203/376448
Soyka Datentechnik, Hattinger Str. 685, 4630 Bochum 5, Tel. 0234/411913
AB Computersysteme, A. Büdenbender, Wildenburgstr. 21, 5000 Köln 41, Tel. 0221/4301442
C.A.S., Sprendlinger Landstr. 71, 6050 Offenbach, Tel. 069/842013
Jumbo Soft, Horemansstr. 2, 8000 München 19, Tel. 089/1234065

Top aktuell:









3-D-Grafik auf dem Amiga – hier ist der Schlüssel zu dieser faszinierenden Welt. In diesem Buch werden Grafikalgorithmen beschrieben und erläutert, die es in sich haben. Mit ihnen können Sie absolut realistisch gestaltete Bilder erstellen: Die einzelnen Komponenten eines Bildes lassen sich dabei mit einem Editor problemlos eingeben und solange durch die Wahl verschiedener Materialien, Farben, Lichtquellen und Spiegelungen verfeinern, bis sie eine absolut naturgetreue Darstellung erreicht haben. Die Berechnung des Bildes übernimmt das Programm automatisch, in allen Auflösungen mit bis zu 4096 Farben. 3-D-Grafikprogrammierung – ein Muß für alle, die Spaß an tollen Grafiken haben.
Amiga 3-D-Grafikprogrammierung
Hardcover, 283 Seiten
inkl. Diskette, DM 59,–

Grafik ist das zentrale Thema des Amiga. Und mit dem Supergrafik-Buch können Sie hier in Dimensionen vorstoßen, die Sie selbst einem Amiga nicht zutrauen würden. Aber keine Sorge: Amiga Supergrafik wurde nicht nur für Profis geschrieben. Es bietet jedem etwas. So gibt es zahlreiche Einsteiger-Programme, die das nötige Grundwissen vermitteln, ebenso wie Programme für den fortgeschrittenen BASIC-Programmierer. Der Profi hingegen erfährt, wie er die Grafik von C aus ansprechen kann. Kurzum ein Buch, in dem Sie finden, was Sie brauchen: Grafikprogrammierung mit den vorhandenen BASIC-Befehlen, Nutzung der Libraries, die Register der Grafik-Chips, Aufbau und Programmierung von Screens, Windows, HAM, Halfbrite und Interlace aus BASIC und C, 1024 x 1024-Superbitmap, gepufferte Multitasking-Hardcopy-Routine – zum Thema Grafik werden Sie in Amiga Supergrafik nichts vermissen.
Amiga Supergrafik
686 Seiten, DM 59,–

Der Amiga ist das ideale Werkzeug für Kreative – aber einen Film mit ihm erstellen? Warum nicht? Dieses Buch hilft Ihnen dabei. Vom Drehbuch bis zum fertigen Drei-Minuten-Film. Mit allem, was dazu gehört: Grafik, Sound, Animation und und und. Dabei arbeiten Sie mit den verschiedenen DeLuxe-Programmen – auch mit DeLuxe-Video 1.2 und DeLuxe-Paint II, lernen deren fantastische Möglichkeiten und Anwendungsgebiete kennen. Da Sie als angehender Filmemacher auch mit Videorecorder und Sounddigitizer professionell umgehen müssen, wird das nötige Know-how gleich mitgeliefert. Wann dürfen wir zu Ihrem ersten Oskar gratulieren?
Amiga – Der Film
Hardcover, 407 Seiten
DM 49,–

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:
zzgl. DM 5,– Versandkosten
unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei
Name
Straße
Ort

Tausendsassa mit Tücken

AMIGA test **Auf dem PC schon längst ein Erfolg, schickt sich Word Perfect nun an, neue Maßstäbe für Amiga-Textverarbeitungen zu setzen. Hat das Programm eine Chance?**

Commodores Präsident Irving Gould ist von der neuen Textverarbeitung merklich begeistert. »Amiga-Besitzer, die Word Perfect noch nicht kennen, werden von den hervorstechenden Eigenschaften des Programms beeindruckt sein, die es von anderen Produkten dieser Klasse abheben«, schrieb er an Alan C. Ashton, den Präsidenten der im amerikanischen Bundesstaat Utah ansässigen Word Perfect Corporation.

Besonders positiv bewertete der Commodore-Chef die »Ergänzung neuer Funktionen, die von den einzigartigen Fähigkeiten des Amiga vollen Gebrauch machen, wobei gleichzeitig all jene Qualitäten beibehalten wurden, die Word Perfect bei den PC-Usern so beliebt gemacht haben«.

Bei so viel Lob durfte man natürlich gespannt sein, ob mit Word Perfect tatsächlich eine neue Ära der Textverarbeitung auf dem Amiga angebrochen ist. Und weil wir unmöglich warten konnten, bis irgendwann in (hoffentlich nicht so ferner) Zukunft die deutsche Version veröffentlicht wird, haben wir uns das englische Original von Word Perfect besorgt und für Sie gründlich unter die Lupe genommen.

Schon äußerlich präsentiert sich das Programm als Schwergewicht: Exakt 1,95 Kilogramm bringt der stabile Kartonschuber auf die Waage, in dem sich zusätzlich zu Software, Handbuch und einigen nützlichen Accessoires noch ein Formular befindet, mit dem sich der Käufer als legitimer Benutzer registrieren lassen sollte, um in den Genuß des vorbildlichen Kundendienstes zu kommen. Er darf dann nicht nur auf Updates hoffen, sondern kann sich auch bei etwaigen Problemen direkt, über die sogenannte »Hotline«, telefonisch beim Hersteller Rat holen — ein Beispiel, das Schule machen sollte.

Vorbildlich ist auch das englische Manual, das als praktischer Ringhefter gehalten ist und auf mehr als 600 Seiten Aufschluß über die Fähigkeiten von Word Perfect gibt. In Verbindung mit einer der vier Programmdisketten bietet das Handbuch im Kapitel »Learn« sogar einen didaktisch gut aufgebauten Schritt-für-Schritt-Einführungskurs, der blutigen Anfängern ebenso anzuraten ist wie erfahrenen Textverarbeitern; denn es bedarf schon einiger Übung, um die zahlreichen Möglichkeiten des Programms, auf die wir noch zu sprechen kommen werden, wirklich ausnutzen zu können. Nicht umsonst heißt es im Handbuch vielsagend: »Verlieren Sie nicht den Mut, wenn Sie nicht verstehen, was auf dem Bildschirm vor sich geht,

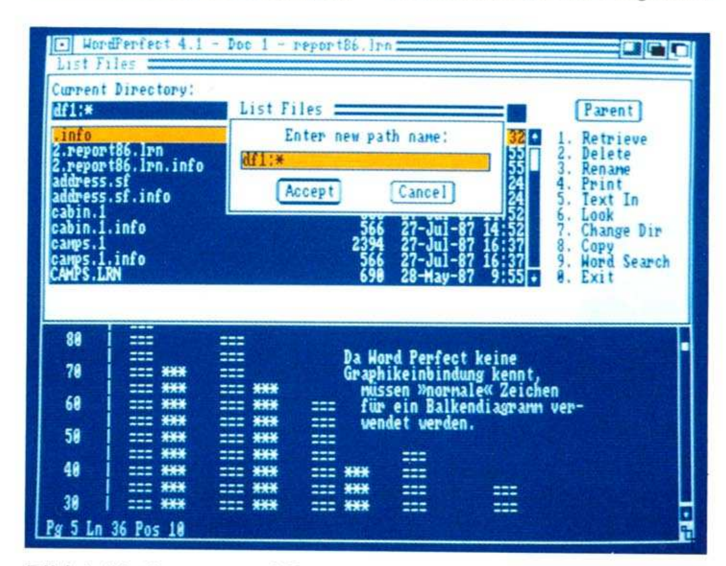


Bild 1. Listfenster und Requester für den Dateitransfer

oder wenn Sie eine Lektion wiederholen müssen. Es braucht seine Zeit, sich daran zu gewöhnen, wie Word Perfect und Ihr Computer zusammenarbeiten.«

Ungewöhnlich für ein amerikanisches Programm ist der fehlende Kopierschutz, der sowohl die problemlose Installation von Word Perfect auf einer Festplatte als auch das Anfertigen von Arbeitskopien ermöglicht. Da man ohne Anleitung nicht viel mit dem Programm anfangen kann, erübrigt sich ein Kopierschutz ohnehin.

Lobenswert auch, daß der Hersteller von vornherein auf größtmögliche Internationalität seines Produkts geachtet hat. Dadurch kann Word Perfect nicht nur den PAL-Bereich des Bildschirms voll ansteuern, es gibt sich auch in puncto Fremdsprachen sehr flexibel: Ein Wechsel des Tastaturtreibers — zum Beispiel mit »setmap d« im CLI — genügt, und schon arbeitet das Programm ohne Fehl und Tadel mit den deutschen Umlauten zusammen — nur die Pull-down-Menüs bleiben natürlich englisch. Darüber hinaus ist das Programm für individuelle Wünsche des Anwenders sehr aufgeschlossen und erlaubt durch das »Set up«-Menü eine jederzeit veränderbare Festlegung vieler Voreinstellungen wie etwa Tabulatorpositionen oder die Formularlänge.

Spartanisches Textfenster

Wäre der erste Eindruck tatsächlich immer der entscheidende, sähe Word Perfect wohl wenig rosigen Zeiten entgegen. Eher kärglich und ganz ohne die grafischen Beigaben, die man von Amiga-Programmen sonst gewohnt ist, gleicht die Arbeitsfläche des Programms fast schon einem Workbench-Fenster. Einzig ein ungewohntes Gadget in der rechten oberen Ecke — mit dem ein verkleinertes Textfenster blitzschnell wieder auf die volle Größe gebracht werden kann — und die obligatorische Statuszeile am unteren Bildrand zeigen den Unterschied an. Allerdings reicht diese bei weitem nicht aus, um den Anwender über alle wichtigen Einstellungen auf dem laufenden zu halten, denn sie informiert in der Regel lediglich über Seitenzahl und Cursorposition. Schon grundlegende Informationen wie Textbreite, Format (links-, rechtsbündig, zentriert, Blocksatz) oder Tabulatorpositionen gibt Word Perfect erst auf zum Teil mehrfachen Tastendruck hin preis.

Wie andere Amiga-Textsysteme auch, beherrscht das Programm die Mehrfensterverarbeitung, das heißt, es können so viele Dokumente gleichzeitig bearbeitet werden, wie es der Arbeitsspeicher zuläßt. Sollte dieser jedoch nur 512 KByte umfassen, erscheint schon beim ersten Versuch, ein weiteres Textfenster zu öffnen, der Hinweis »Volume RAM is full«. Auch um längere Texte vernünftig bearbeiten zu können, ist eine Speichergröße von mindestens einem MByte unabdingbar. Ferner greift Word Perfect öfter auf die Bootdiskette zurück, so daß ohne ein zweites Diskettenlaufwerk — welches seinerseits ebenfalls rund 30 KByte RAM beansprucht — nicht viel auszurichten ist.

Als Merkhilfe legt der Hersteller zwar eine Tastaturschablone und eine Referenzkarte bei, aber dennoch vergeht viel Zeit, bis einem die Funktionstastenbelegung »ins Blut« übergegangen ist. Da über die Dreifachbelegung der F-Tasten nur 30 Funktionen angesteuert werden können, mußten die Programmierer die meisten Möglichkeiten von Word Perfect in Menüs, Untermenüs und Unteruntermenüs ansiedeln, um die vollständige Bedienbarkeit mittels der Tastatur zu gewährleisten. Hinzu kommt, daß auf die »Shortcuts« zwar nicht völlig verzichtet wurde, der Anwender sich aber von dem gewohnten Verfahren (<Amiga>-<Buchstabe>) lösen muß. Blockoperationen etwa werden bei Word Perfect über die Tastenkombination <Ctrl>-<F4> ausgelöst. Dadurch gerät man aber erst in das Menü »Move«, in dem man nun näher bestimmen muß, welche Textbewegung ausgeführt werden soll. Bei anderen Programmen lassen sich die

wichtigsten Blockoperationen (Ausschneiden, Kopieren, Einfügen) durch die Shortcuts schnell und einfach abwickeln, während kompliziertere Textbewegungen oft nur mit der Maus aktiviert werden können.

Bei Word Perfect gibt es diese Unterteilung nicht, so daß man immer erst ins »Move«-Menü muß, um dort näher zu bestimmen, was mit dem markierten Textbereich geschehen soll. Allerdings gibt es eine Möglichkeit, sich für häufig gebrauchte Befehlsfolgen eigene Shortcuts zu gestalten, doch dazu später Genaueres. Daß diese umständliche Bedienung durch pure Gedankenlosigkeiten im Programmkonzept noch verstärkt wird, zeigt das Beispiel Seitennumerierung: Word Perfect bietet hierfür eine Reihe von Möglichkeiten, die nach dem Anwählen von »Page Number Position« im »Print Format«-Menü aufgelistet werden. Per Tastendruck oder Mausklick hat der Anwender nun die freie Auswahl. Auch wenn das auf den ersten Blick bequem erscheinen mag — tatsächlich ist es umständlich. Bei anderen Amiga-Textverarbeitungen gestaltet man seine Kopf- und Fußzeilen so, wie sie beim Endprodukt aussehen sollen, und trägt für die Seitennummer an entsprechender Stelle einfach einen Platzhalter — oftmals ein Doppelkreuz (#) — ein. Das hat nicht nur den Vorteil der größeren Flexibilität, es erspart dem Anwender auch ein weiteres Menü, und sorgt überdies dafür, daß die Pull-down-Menüs nicht, wie bei Word Perfect geschehen, vor lauter Einzelpunkten schier überquellen.

Wenig bedienungsfreundlich präsentiert sich das Programm auch, wenn es darum geht, ein Dokument einzuladen oder es zu speichern. Hier tut sich schlicht eine kleine Dialogbox auf, in die der Filename eingetragen wird — was sich sonst noch auf der Diskette befindet, wird erst nach dem Anwählen von »List Files« gezeigt.

Dateiauswahl mit Komfort

Dann allerdings gibt Word Perfect eine erste Kostprobe seiner Funktionenvielfalt, denn das Diskettenfenster (Bild 1) bietet weit mehr als Standardfunktionen wie Inhalt anzeigen und Dokumente laden. Will man etwa nachsehen, ob sich ein bestimmter Text auf der eingelegten Diskette befindet, so tippt man einfach seinen Namen, und meistens reichen dem Programm schon die Anfangsbuchstaben, um das File aus dem Inhaltsverzeichnis herauszusuchen. Interessant ist auch die Möglichkeit, die gesamte Diskette nach einem bestimmten Text durchsuchen zu lassen. Nehmen wir an, Sie suchen alle Dokumente, in denen das Wort »Amiga« mindestens einmal vorkommt. Statt nun sämtliche in Frage kommenden Texte nacheinander einzuladen und mit der Text-Suchfunktion zu überprüfen, wählen Sie »Search« im Diskettenfenster und geben den gesuchten Begriff ein. Word Perfect geht daraufhin sämtliche Files der Diskette durch und gibt Ihnen danach an, in welchen Dokumenten das Wort »Amiga« gefunden wurde.

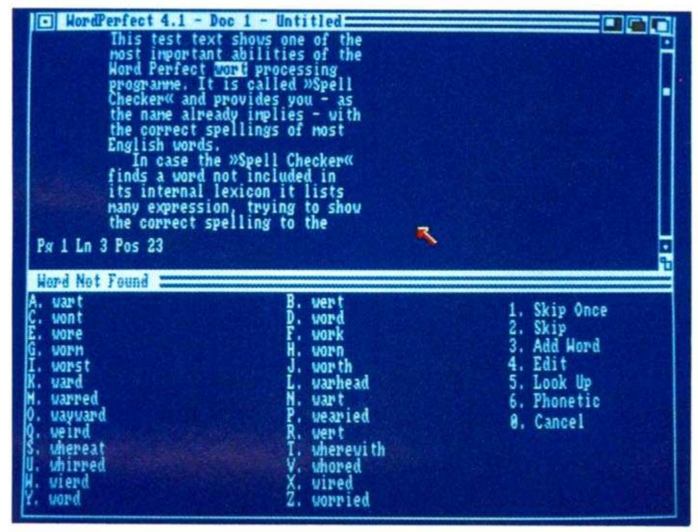


Bild 2. Word Perfect hilft bei der Rechtschreibung

Als sehr nützlich hat sich auch die Funktion »Look« erwiesen, mit der man in Files, unter deren Namen man sich nicht mehr so recht etwas vorstellen kann, mal eben »hineinschnuppern« kann, ohne sie gleich einladen zu müssen. Das gilt allerdings nicht für Word-Perfect-Dokumente, die mit einem Paßwort gesichert wurden. Negativ macht sich in diesem Programmteil lediglich bemerkbar, daß bei jedem Anwählen von »List Files« erst der in einem Requester erscheinende Pfadname bestätigt werden muß, selbst wenn er gar nicht verändert wurde.

Zugegeben, als Amiga-Besitzer ist man etwas verwöhnt, wenn es um die weitestgehende Übereinstimmung von Textaussehen auf dem Bildschirm und tatsächlichem Ausdruck geht. WYSIWYG (»What You See Is What You Get«) lautet die etwas skurrile Bezeichnung für diese Eigenschaft einer Textverarbeitung, die bei manchen Programmen stärker, bei anderen weniger deutlich ausgeprägt ist. Was allerdings Word Perfect in dieser Hinsicht leistet, wird vor allem den grafischen Fähigkeiten des Amiga keineswegs gerecht. Kaum mehr als die Schriftattribute »fett«, »kursiv« und »unterstrichen« wird bereits auf der Mattscheibe sichtbar. Schon bei »Super«- und »Subscript« wird die Schriftänderung nur noch in der Statuszeile angezeigt.

Eingebautes Lexikon

Ähnlich trostlos sieht es bei der restlichen Textformatierung aus: Zentrieren und rechtsbündig Schreiben, das geht schon auf dem Bildschirm. Aber bei fast allen übrigen Gestaltungsmöglichkeiten (selbst beim Blocksatz) verschafft nur ein Probeausdruck endgültig Klarheit — das mutet an wie ein Rückfall ins Computermittelalter. Merkwürdigerweise machen sich diese Einschränkungen aber kaum positiv im Hinblick auf die Arbeitsgeschwindigkeit bemerkbar. Das Programm läßt sich zwar beim Scrolling nicht gar so viel Zeit wie »Vizawrite« oder das am PC verwendete »Word«, kann aber mit einem Tempo, wie es etwa »Textomat Amiga« vorlegt, auch nicht mithalten. Deshalb ist es schade, daß Word Perfect nicht im Interlace-Modus betrieben werden kann, um die Scrolldauer in Grenzen zu halten.

Tippfehler sind von jeher ein Ärgernis beim Schreiben, auch wenn das Hantieren mit »Tipp-Ex« und ähnlichen Chemikalien bei der Textverarbeitung zum Glück entfällt. Gerade bei langen Texten, die man wieder und wieder verändert, bleibt es aber meist nicht aus, daß trotz aller Wachsamkeit des Verfassers einige falsche Buchstaben ihren Weg aufs Papier finden und erst dort — wenn überhaupt — entdeckt werden. Das ist natürlich ärgerlich, zumal es meist bei Texten auftritt, denen Bedeutung beigemessen wird (zum Beispiel eine Bewerbung).

Hier nun kommt eine jener Möglichkeiten ins Spiel, die Word Perfect bei PC-Besitzern Popularität verschafft haben: das eingebaute Lexikon, das dem Anwender die lästige Tippfehler-Suche vollständig abnimmt. Dieser »Spell Checker« befindet sich auf einer eigenen Diskette und wird bei Bedarf teilweise nachgeladen. Daher ist diese Funktion nur bei »hochgerüsteten« Amigas mit mindestens 1 MByte sinnvoll einsetzbar.

Die Überprüfung des Textes kann für ein Wort, eine Seite oder für das ganze Dokument durchgeführt werden und geht sehr flott vonstatten, solange Word Perfect jeden Begriff in seinem Lexikon wiederfindet. Ist das nicht der Fall, halbiert sich das Textfenster und macht Platz für das der Rechtschreibkontrolle, in dem nun Alternativen zu dem unbekannten Begriff aufgelistet werden (Bild 2). Weil das Programm dazu auf die Diskette zurückgreift, kann es einige Zeit dauern, bis für ein falsch getipptes Wort die korrekte Schreibweise angezeigt wird. Speichermillionären ist deshalb zu empfehlen, sich im CLI das Lexikon ins RAM zu kopieren (mit »copy Spell.Thes:lex.wp to ram:«). Damit dann nicht auch weiterhin auf der Diskette nachgesehen wird, muß zusätzlich der Pfadname für das Lexikon entsprechend verändert werden (»ram:lex.wp«). Die Rechtschreibkontrolle dezimiert den Arbeitsspeicher zwar um fast 300 KByte, aber wenn nicht gerade mehrere Textfenster gleichzeitig geöffnet sind, kann man das gut verschmerzen. Vor allem, da der Suchvorgang um ein Mehrfaches beschleunigt wird. Wurde das beanstandete Wort korrekt geschrieben und soll deshalb weder durch einen von Word Perfect vorgeschlagenen Begriff ersetzt noch korrigiert werden, kann man es entweder ins Le-

xikon aufnehmen oder schlicht überspringen.

Sehr nützlich ist auch die Möglichkeit, ein Wort nachschlagen zu können beziehungsweise nach phonetischen Gesichtspunkten suchen zu lassen. Dabei achtet das Programm nicht mehr vornehmlich auf die Schreibweise des Suchbegriffs, sondern auf dessen Aussprache. Gibt man beispielsweise die Buchstabenkombination »eez« ein, so findet Word Perfect als Alternative das englische Wort für »Leichtigkeit«: »ease«, denn beides klingt im angelsächsischen Sprachraum gleichermaßen wie »iis«.

Auf der gleichen Diskette wie der Spell Checker befindet sich ein Programm mit dem Namen »Thesaurus«, und obwohl das so ähnlich klingt wie »Tyrannosaurus«, hat es nichts mit urzeitlichen Riesenechsen zu tun. Vielmehr ist es der englische Ausdruck für ein Synonym-Wörterbuch (im Deutschen zum Beispiel der »Duden der sinn- und sachverwandten Wörter«).

Vielschreiber wissen ein solches Werk zu schätzen, da sie, um Wiederholungen zu vermeiden, oftmals auf Synonyme zurückgreifen müssen — und wer hat die schon alle auf Anhieb parat? Bei Word Perfect erübrigt sich sogar das Nachschlagen, da der Thesaurus ebenso wie die Rechtschreibkontrolle bei Bedarf eingeladen wird und binnen Sekunden arbeitsbereit ist.

Es lohnt sich allerdings nicht, diese ebenfalls sehr speicherintensive Funktion ins RAM zu kopieren, da die Suche auf Diskette hier viel zielgerichteter und somit schneller verläuft als beim Spell Checker. Der dabei nötige Diskettenzugriff sorgt nicht für eine spürbare Verzögerung.

Fast schon ein Duden

Das Wörterbuch arbeitet nach einem einfachen, aber sehr effektiven Prinzip. Wiederum wird zuerst das Textfenster horizontal halbiert, um im unteren Bildschirmbereich den Platz zu schaffen, in dem die Synonyme zweispaltig aufgelistet werden können (Bild 3). Nach der Eingabe des Suchwortes gibt Word Perfect alle korrespondierenden Begriffe aus, und zwar säuberlich unterteilt nach den verschiedenen Wortarten (Substantiv, Verb, Adjektiv).

In dieser Liste sind alle sogenannten »headwords« mit einem Punkt markiert, das heißt für diese Wörter kann das Programm auf Wunsch ebenfalls Synonyme nennen. Diese werden dann in der rechten Spalte aufgelistet. Dort kann man ebenfalls wieder »Kopfwörter« anklicken und sich so immer weiter durch das Wörterbuch hindurcharbeiten, bis man einen passenden Begriff findet.

Weil man sich dabei aber unter Umständen recht schnell recht weit von der ursprünglichen Bedeutung des Suchwortes entfernt, bleibt die linke Spalte immer auf dem Bildschirm präsent. Dadurch kann der Anwender auch nach längerer ergebnisloser Suche noch ein neues, unmittelbar zum Suchbegriff korrespondierendes »headword« direkt anwählen, ohne zuvor den Suchbegriff selbst ein zweites Mal eingeben zu müssen.

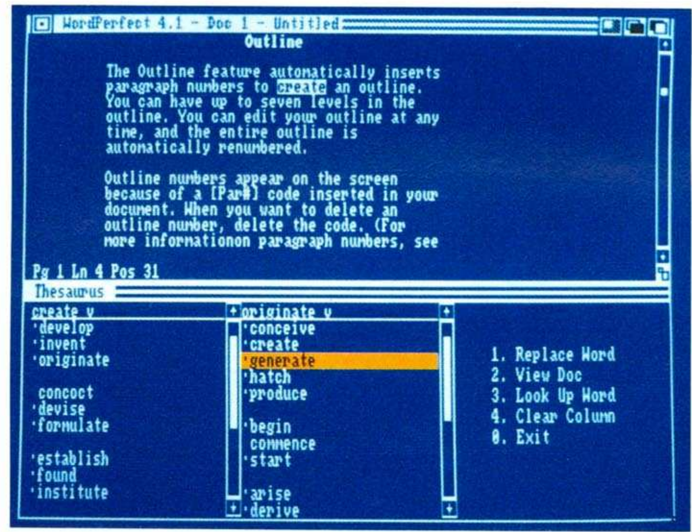


Bild 3. Der »Duden« für sinn- und sachverwandte Worte

Beide Lexika, sowohl der Spell Checker als auch der Thesaurus, sind zum gegenwärtigen Zeitpunkt freilich allenfalls für Englisch-Studenten interessant, zumal die Rechtschreibkontrolle zum Systemabsturz führt, sobald sie auf einen Umlaut stößt. Deshalb ist es auch nicht möglich, sich mit dem englischen Word Perfect selbst ein deutsches Lexikon zu erstellen. Sollten diese Funktionen aber in gleichem Umfang auch in der deutschen Version des Programms implementiert sein — und es gibt nichts, was dagegen spräche —, sind dies in der Tat zwei gewichtige Pluspunkte für Word Perfect.

Erinnern Sie sich noch an die Anfänge der Textverarbeitung, als der Anwender die meisten Druckerkommandos direkt in den Text eingeben mußte, so daß das Auge beim Lesen jedesmal aufs Neue ins Stolpern kam? Wahrscheinlich sind Sie froh, daß diese Zeiten vorbei sind. Wer mit Word Perfect ernsthaft arbeiten will, wird an diesen Funktionscodes allerdings nicht ganz vorbeikommen, sind sie doch ein essentieller Bestandteil des Programmkonzepts. Freilich verunstalten sie, wie bei allen modernen Textsystemen, nicht mehr unmittelbar das Geschriebene, aber man kann sie in einem eigenen Fenster

Codes stiften Verwirrung

sichtbar machen — und ab und zu läßt sich das auch gar nicht vermeiden. Das ist etwa der Fall, wenn der Drucker Dinge zu Papier bringt, die auf dem Bildschirm gar nicht zu sehen waren. Zwar passiert das selten, kommt aber besonders dann vor, wenn ein »Center«-Code nicht dort gesetzt wurde, wo er hingehört.

Generell fügt Word Perfect für jede Formatänderung einen Funktionscode in den Text ein. Diese Definition gilt immer ab der jeweiligen Cursorposition, und zwar so lange, bis sie durch einen anderen Code aufgehoben wird. Das kann von Vorteil sein, weil man nicht für jede Änderung ein störendes Lineal einzufügen braucht. Es kann sich aber auch nachteilig auswirken, wenn man eine Änderung für den gesamten Text vornehmen will, es aber versäumt, den Cursor zuvor an den Anfang zu setzen. In diesem Zusammenhang macht sich auch immer wieder das fehlende WYSIWYG bemerkbar. Außerdem wird derselbe Code bei manchen Funktionen unsinnigerweise mehrfach eingefügt, wenn man diese mehrmals nacheinander aufruft. Das gilt beispielsweise für die Randeinstellung, aus deren Dialogbox man nach zwei Cursorbewegungen quasi »herausgeworfen« wird, etwa wenn man nach dem Einstellen des rechten Randes noch einmal den linken ändern möchte. Dann muß man diese Dialogbox erneut anwählen, und Word Perfect fügt den Code erneut in den Text ein.

Angesichts des eingebauten Lexikons mutet es etwas gedankenlos an, daß die Programmierer Word Perfect keine automatische Silbentrennung spendierten, schließlich wäre es ein leichtes gewesen, zusätzlich zu den Wörtern selbst noch deren mögliche Trennungen im Spell Checker festzuhalten.

Statt dessen steht dem Anwender nur eine äußerst unzureichende Trennhilfe zur Verfügung, die immer dann aktiviert wird, wenn ein Wort zu weit links vom rechten Rand beginnt und zusätzlich über diesen hinausragt. Die Breite dieser Zone kann frei definiert werden und richtet sich danach, wie stark das »Ausfransen« des Randes ausfallen soll. Um diesen unerwünschten Effekt so gering wie möglich zu halten, wählt man am besten eine Breite von zwei bis drei Zeichen.

Bei jedem längeren Wort öffnet sich dann eine Dialogbox, in der der Trennstrich mit der rechten und der linken Cursortaste im Wort verschoben werden kann. Da dieser Vorgang natürlich den Schreibfluß stört, empfiehlt es sich, die Trennfunktion erst nach Vollendung des Textes zu aktivieren.

Unverständlich bleibt allerdings, warum der Trennstrich innerhalb eines Wortes nicht unbegrenzt weit nach links bewegt werden kann. Bei langen Worten kommt es daher hin und wieder vor, daß man erst gar nicht zu der Silbe gelangt, nach der getrennt werden könnte. Statt dessen muß man den Begriff vollständig in die neue Zeile übernehmen.

Ärgerlich auch, daß die Trennfunktion bei unserer Testversion häufig zum Systemabsturz führte. Der äußerte sich allerdings nicht als »Guru«, sondern darin, daß der Cursor auf Nimmerwiedersehen verschwand und nur noch der Mauszeiger bewegt werden konnte. Erst nach Drücken der

Sie haben einen Amiga, Atari ST oder QL? Wir haben die Zeitschrift dazu!

Markt&Technik

9/87 **68000er** DAS MAGAZIN DER NEUEN COMPUTER-GENERATION SEPTEMBER 1987

DM 7,–

Grafik
- Test: die besten Programme
- Einführung: meisterhaft malen
- Programmierung: Blitter

Amiga
- Zum Mega-Amiga aufrüsten
- Tolles DFÜ-Duo

Atari ST
Tempo für Basic: Neuer Compiler

QL
Hardware klar und verständlich

Viele anspruchsvolle Listings zum Abtippen

Amiga, Atari ST, QL — die Spitzentechnologie im »68000er«

»68000er«, das Magazin der neuen Computer-Generation, bringt ausführlich Berichte, Tests und Beschreibungen von

- Hard-, Software und Peripherie
- Programmiersprachen: Kurse in Basic, C, Modula 2, Assembler und Pascal
- Homeentertainment: aktuelle Spiele, neue Grafikprogramme
- Tips & Tricks und Bauanleitungen für Hardware-Erweiterungen

Machen Sie sich die Technologie des Amiga, Atari St und QL sofort verfügbar – nutzen Sie die Chance, mit einem kostenlosen Probeexemplar das »68000er«-Magazin kennenzulernen.

Markt&Technik Verlag
Aktiengesellschaft
Hans-Pinsel-Straße 2
D-8013 Haar bei München



KENNENLERN-ANGEBOT

MIT EINEM KOSTENLOSEN »68000er«-PROBEEXEMPLAR

JA, ich möchte »68000er«, das Magazin der neuen Computer-Generation, kennenlernen.
Senden Sie mir bitte die aktuelleste Ausgabe kostenlos als Probeexemplar. Wenn mir »68000er« gefällt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte es dann regelmäßig frei Haus per Post. Außerdem nutze ich den Abonnement-Preisvorteil von 8% und bezahle pro Jahr nur 77,– DM statt 84,– im Einzelverkauf.

Name, Vorname

Straße

PLZ, Ort

Datum, 1. Unterschrift

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestelladresse widerrufen kann und bestätige dies durch meine zweite Unterschrift. Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs.

Datum, 2. Unterschrift.

Gutschein ausfüllen und absenden an: Markt & Technik Verlag Aktiengesellschaft, Vertrieb, Postfach 1304, 8013 Haar bei München.

A188

Menütaste rührte dieser sich ebenfalls nicht mehr vom Fleck, und es blieb lediglich der Reset als letzter Ausweg. In Verbindung mit ASCII-Files, die von Diskette geladen wurden, trat dieser Fehler sogar grundsätzlich auf. Gerade im Falle derartiger »Hänger« ist es ärgerlich, wenn man den Text gar nicht oder lange vor dem Mißgeschick letztmals abgespeichert hatte. Da manch schöpferische Natur im Eifer des Gefechts schon mal zur Vergeßlichkeit neigt, bietet Word Perfect die Möglichkeit, in bestimmten Zeitintervallen automatisch Sicherheitskopien, sogenannte Backups, zu erstellen. Sobald man das Programm ordnungsgemäß verläßt, wird dieses File automatisch gelöscht, und zwar auch dann, wenn man selbst das Dokument nicht mehr gespeichert hat. Wer darauf nicht achtet, verliert letztlich also doch noch die Früchte seines Schaffens.

Ein Schwachpunkt vieler Amiga-Textverarbeitungen ist die Druckeransteuerung, die sich häufig darauf beschränkt, die Systemroutine »Printer.device« anzusteuern und alles andere dem Rechner selbst zu überlassen, so daß Geräte, die nicht Preferences-kompatibel sind, Verständigungsschwierigkeiten haben und schlimmstenfalls »Datensalat« zu Papier bringen. In dieser Hinsicht setzt Word Perfect neue Maßstäbe. Bereits die eigenen mitgelieferten Treiber (sage und schreibe 251 an der Zahl) reizen wirklich jede Möglichkeit des entsprechenden Geräts aus. Durch einige Tricks bei den Befehlssequenzen werden selbst 9-Nadel-Drucker in die Lage versetzt, einen Text in NLQ zu schreiben und dennoch sämtliche Funktionen zu nutzen, die vielfach nicht mit der Schönschrift kombinierbar sind (etwa Kursivschrift oder Super/Subscript). Dazu wird dann jedesmal erst die NLQ ausgeschaltet, bevor das Kommando etwa für Superscript kommt. Nach Beendigung dieser Schriftart aktiviert der Word-Perfect-Treiber die NLQ dann wieder, so daß stets die bestmögliche Druckqualität gewährleistet ist.

Besitzer ganz neuer, ganz alter oder ganz exotischer Geräte können sich bei Bedarf auch einen eigenen Druckertreiber erstellen. Das entsprechende, gut dokumentierte und sehr umfangreiche Programm befindet sich auf einer eigenen Diskette, die mitgeliefert wird.

Negativ fällt beim Ausdruck auf, daß dazu jedesmal erst das Unterprogramm »Print« und der gerade aktivierte Druckertreiber von der Word-Perfect-Diskette geladen werden müssen. Auch ist nicht ersichtlich, wie die acht Druckerfonts, die für jeden Treiber voreingestellt wurden, letztlich aussehen. So ist Font drei etwa meist eine Proportionalschrift, nur ist das vom Programm selbst nicht zu erfahren. Auch hier muß also wieder fleißig ausprobiert werden. Ferner läßt sich ein einmal begonnener Druckvorgang nicht jederzeit abbrechen.

```
WordPerfect 4.1 - Doc 1 - spaltentext.lrn
Dies sind Spalten in »newspaper-style«:

     Fish Tail Shore                      White Sands Beach

Fish Tail Shore was named for        White Sands Beach was named
its natural shape.  Over the         for its unusually brilliant
ages, sand deposits have built       sand.  The area is ideal for
up from both sides of the            swimming, volleyball, and
point creating a natural             building sand castles.
harbor.  This campsite is            Fishing is not allowed on this
located on the north side of         beach.  This campsite is
the reservoir.                       located on the west side of
                                     the reservoir.

Für Adreßlisten eignen sich besonders gut parallele
Spalten, die Word Perfect automatisch einander angleicht.

Ranger Hyrum R. Stone     Reservoir Campgrounds     801 555-6666
                          Park Headquarters
                          Lone Pine National Park
                          Flatwood, Utah 84999

Ranger John Bigney        Silver Mines Flats        801 555-4444
                          Ranger Station
                          Lone Pine National Park
                          Flatwood, Utah 84999

Ranger Jessie Parker      Needle Rock               801 555-3333
                          Ranger Station
 Pg 1 Ln 16 Pos 67
```

Bild 4. Word Perfect verarbeitet mehrspaltige Texte

Als erstes Amiga-Textsystem bietet Word Perfect eine automatische Fuß- und Endnotenverwaltung, die besonders beim Erstellen wissenschaftlicher Texte eine sehr große Hilfe darstellt; denn sie sorgt dafür, daß jede Fußnote immer am Ende der betreffenden Seite steht, egal wie oft das Textformat geändert wird. Endnoten werden fortlaufend numeriert und zusammen auf einem eigenen Blatt aufgeführt.

Die automatische Erstellung von Indizes und Inhaltsverzeichnissen ist für Word Perfect ebenfalls kein Problem — eher schon für den Anwender, denn die Bedienung ist wiederum recht umständlich. Nicht nur hier machen sich die sogenannten Makros bezahlt. Das sind ganze Befehlssequenzen oder auch Floskeltexte, die als File auf der Programm-Diskette abgespeichert werden und bei Bedarf blitzschnell eingeladen werden. Wenn man nicht gerade Tastenkombinationen wie <Amiga>-<P> für »Text ausdrucken« benutzt, sondern den Makros »richtige« Namen gibt, verliert man allerdings schnell die Übersicht und muß sich die Makro-Bezeichnungen auf einem Zettel notieren oder im »List Files«-Menü nachgucken — das kostet Zeit.

Zum leichten Erstellen von Adressenlisten oder Tabellen bietet Word Perfect die sehr nützliche Möglichkeit, Text mehrspaltig zu formatieren (Bild 4). In Zahlenspalten lassen sich zudem die vier Grundrechenarten, die das Programm ebenfalls beherrscht, sehr vielfältig einsetzen.

Erwähnt werden soll schließlich noch die Serienbrieffunktion, die außerordentlich leistungsfähig ist und sich unter anderem auch gut dazu eignet, Standardformulare zu erstellen. Aus Platzgründen ist es leider nicht möglich, hier alle Funktionen von Word Perfect aufzuzählen. Oft handelt es sich dabei zwar nur um kleine, aber dafür nützliche Details.

Lohnt sich der Kauf?

Es fällt schwer, ein pauschales Urteil über Word Perfect zu sprechen. Zweifellos ist es das umfangreichste Textverarbeitungsprogramm, das bisher für den Amiga präsentiert wurde, und steht in puncto Leistungsfähigkeit somit konkurrenzlos da. Andererseits ist die Bedienung des Programms sehr gewöhnungsbedürftig, und gerade im Hinblick auf die geradezu unüberschaubar zahlreichen Möglichkeiten muß sich der Hersteller fragen lassen, warum er sich konzeptionell für die Tastaturbedienung entschieden hat, die praktisch ein tägliches, intensives Arbeiten mit dem Programm voraussetzt und einen längeren Urlaub oder ähnliche Unterbrechungen nicht berücksichtigt. Von Vorteil ist diese Philosophie natürlich für Anwender, die in der Umgebung von IBM-, DEC- oder Siemens-Computern arbeiten und sich nicht auf den Amiga umstellen brauchen.

Verbesserungsbedürftig erscheint uns auch die Textdarstellung auf dem Bildschirm, wo bezüglich WYSIWYG noch einiges getan werden müßte. Last but not least bleibt zu hoffen, daß dem Programm eine automatische Silbentrennung spendiert wird, die bei einer Textverarbeitung dieser Güteklasse nicht fehlen darf. Und dann ist da noch der Preis, den wir bisher höflich verschwiegen haben, obwohl er den Kreis der Word-Perfect-Interessenten beträchtlich einschränken dürfte — schließlich bleiben nach der Anschaffung von Speichererweiterung und zweiter Floppy nicht bei jedem die erforderlichen 800 Mark im Portemonnaie zurück.

(Karsten Lemm/pa)

AMIGA-WERTUNG

Software: Word Perfect (englisch)

7,4 von 12	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	x	x	x	x		
Dokumentation	x	x	x	x	x	x
Bedienung	x	x	x			
Erlernbarkeit	x	x	x			
Leistung	x	x	x	x	x	

Fazit: Preis und Leistungsumfang machen Word Perfect weniger für den gelegentlichen Schreiber, sondern für Verfasser von Artikeln, Referaten, Büchern oder ähnliche Anwendungen interessant. Word Perfect nutzt zwar teilweise den Komfort der Benutzeroberfläche des Amiga, kann aber in dieser Hinsicht noch verbessert werden.

Positiv: Wörterbücher; Serienbrieffunktion; Fußnotenverwaltung; Dokumentation; kein Kopierschutz; Rechenfunktionen

Negativ: keine Silbentrennung; nicht absturzsicher; WYSIWYG nicht durchgehend realisiert

DATEN

Produkt: Word Perfect

Preis: 798 Mark

Hersteller: Word Perfect Corporation

Anbieter: Word Perfect Software GmbH; Soltau Electronics; Atlantis
vollständige Adressen entnehmen Sie bitte dem Anzeigenteil.

Logic Works – simulierte Digitaltechnik

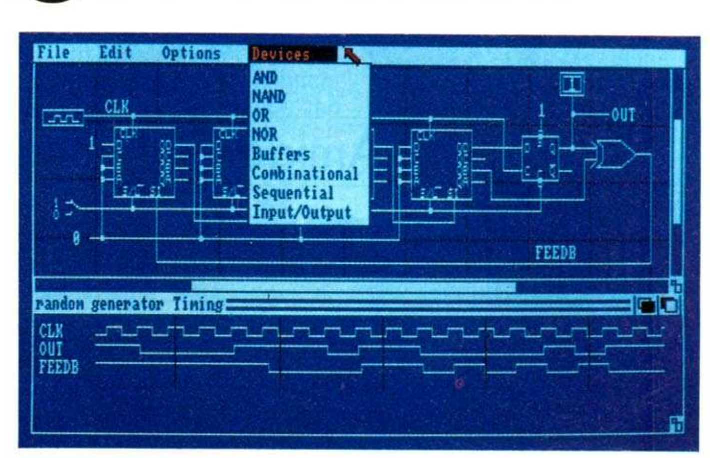
AMIGA test **Der Amiga ist bei weitem nicht nur ein Spielecomputer, sondern auch für Anwendungen zu gebrauchen. Mit »Logic Works« wird das Entwerfen und Testen von Digitalschaltungen zum Kinderspiel.**

In kleinerem Umfang ist auch der Amiga in der Lage, digitale Schaltungen zu simulieren. Möglich wird dies durch ein Programm von Capilano Computing, das sich vorwiegend an mit der Digitaltechnik vertraute Amiga-Besitzer wendet (Bild).

Das Programm ermöglicht die Eingabe von Digitalschaltungen mittels der Maus. In einem Pull-Down-Menü stehen dem Benutzer dabei zahlreiche Grundschaltungen der Digitaltechnik zur Verfügung. Es stehen folgende Schaltungen zur Auswahl:

- Verknüpfungsschaltungen — Hierunter fallen alle Schaltungen, die verschiedene logische Eingangssignale miteinander vergleichen und das Vergleichsergebnis ausgeben.

Neben der Und-Verknüpfung stehen noch eine Oder-Funktion sowie die negierten Formen wie NOR (Nicht-Oder) und NAND (Nicht-Und) zur Auswahl.



Eine Schaltung läßt sich mit beliebigen Takten speisen

- Kippschaltungen (Flip-Flops) — Diese sehr vielfältig einsetzbaren Schaltungen können unter anderem auch zur Datenspeicherung eingesetzt werden.
- Ein/Ausgabeschaltungen — Es ist möglich, digitale Signale mit einem kleinen Display anzeigen zu lassen. Dieses wird einfach an den Ausgang eines Logikbausteins angeschlos-

sen und zeigt von diesem Zeitpunkt die am Ausgang anliegenden digitalen Zustände an. Für größere Anwendungen kann ein kleines Hex-Keyboard zur Eingabe sowie ein Hex-Display zur Ausgabe von Daten benutzt werden.

Insgesamt hat der Anwender über 30 Logikbausteine und Schaltelemente zur freien Verfügung. Besonders interessant dabei ist, daß die meisten Bausteine zu der Logikfamilie 74LSXX von Texas Instruments kompatibel sind.

Problemloser Entwurf

Obwohl »Logic Works« bereits zahlreiche Logik-Bausteine dem Benutzer zur Auswahl bietet, werden Profis dennoch unzufrieden sein. Es ist nicht möglich, eigene, selbstentworfene Schaltsymbole und Bausteine zu definieren.

In einem Pull-Down-Menü kann ein geeigneter Baustein ausgewählt und mit der Maus in das Schaltungs-Fenster eingesetzt werden. Verbindungen zwischen Bausteinen werden einfach mit der Maus gezeichnet. Überraschend beim Aufbau ist die sofortige Anzeige und Reaktion der Schaltung. Die Simulation der Schaltung läuft also ständig, auch wenn die Schaltung noch gar nicht fertig ist. Dies hat den Vorteil, daß Logikpegel sofort bei der Eingabe ersichtlich sind. Im Zap-Mode können Schaltungselemente durch Anklicken entfernt werden. Auch hier werden, sofern sich Ausgabebausteine in der Schaltung befinden, die neuen digitalen Pegel sofort angezeigt. Leider ist es nicht möglich, die zur Auswahl stehenden Elemente zu drehen oder zu spiegeln, um sie in eine Schaltung optimal zu integrieren. Deshalb wird beim Entwurf von Schaltungen oft relativ viel Platz verbraucht. Ein vertikaler und horizontaler Scroll-Balken verhelfen der Software aber zu genügend Platzreserven. Mit einer Textfunktion können alle Signalleitungen benannt werden, die im Timing-Window erscheinen sollen. In diesem Fenster werden ähnlich wie bei einem Oszilloskop die logischen Zustände aller beschrifteten Signalleitungen dargestellt. Für Testzwecke kann in das Timing-Window auch ein komplexer Logikverlauf mit der Maus gezeichnet werden. Die gesamte Schaltung kann jederzeit auch gedruckt werden. Ist sie zu groß für ein Blatt, so wird sie auf mehrere Ausdrucke aufgeteilt, die sich dann zusammenkleben lassen.

Leider haben die Programmierer von Capilano Computing bei wichtigen Dingen mit Bedienungskomfort gespart. So erscheint beim Laden von Schaltungen nur ein kleiner Requester, um den Namen einzugeben. Standardfunktionen wie Cut, Copy und Paste sind zwar bereits im Menü vorhanden, aber noch nicht in das Programm integriert. Ein gestarteter Druckprozeß ist nur durch Ausschalten des Druckers noch zu stoppen. Für Deutschland müßten im übrigen viele Schaltsymbole des Programms angepaßt werden, da sie nicht der deutschen DIN-Norm entsprechen.

Das sehr knapp gehaltene Handbuch gibt nur die nötigsten Informationen. Leider beeinträchtigen einige Mängel den Gesamteindruck des Programmes etwas. Aber die Programmierer von Capilano Computing haben für den nächsten Update schon einige interessante Erweiterungen angekündigt.

(Bernhard Carli/jk)

AMIGA-WERTUNG

Software: Logic Works

6,2 von 12	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	■	■				
Dokumentation	■	■	■			
Bedienung	■	■	■			
Erlernbarkeit	■	■	■	■		
Leistung	■	■	■	■		

Fazit: »Logic Works« ist ein Programm für fortgeschrittene Anwender der Digitaltechnik. Das Programm hat in der Handhabung einige wesentliche Mängel, die aber mit Updates behoben werden sollen.

Positiv: einfache Eingabe der Schaltungselemente; Programmbausteine kompatibel zur SN74LSXX-Serie; Ausdruck auch großer Schaltungen; Darstellung der Timing-Signale.

Negativ: hoher Preis; keine DIN-Norm erfüllt; Inhaltsverzeichnis nicht auflistbar; keine eigenen Bausteine definierbar; Bedienung (außer Eingabe) unkomfortabel.

DATEN

Produkt: Logic Works
Preis: 218 Mark
Hersteller: Capilano Computing
Anbieter: Atlantis, Dunantstr. 53, 5030 Hürth, Tel. 02233/41081

Der Profi-Assembler

AMIGA test

Ohne leistungsfähige Werkzeuge werden die Assembler-Fans früher oder später in C programmieren. Kann der komfortable Assembler Profimat mit seinem Komfort den Umstieg verhindern?

Die Umsteiger vom C 64 werden sich noch an die endlosen Data-Wüsten oder Hexdumps in unserer Schwesterzeitschrift 64'er erinnern. Für die älteren Computer ist Assembler allerdings auch die einzige Alternative für schnelle Programme. Auf dem Amiga haben Assembler durch C einen starken Konkurrenten bekommen. Fast alle Anwendungen lassen sich in C mit nahezu derselben Ablaufgeschwindigkeit realisieren. Warum also noch in Assembler programmieren?

Das erste Argument für diese Art der Programmierung ist sicherlich der Preis der C-Compiler. Der vorgestellte Assembler kostet nicht einmal ein Drittel dessen, was man für einen guten C-Compiler auf den Tisch legen muß. Das stärkste Argument sind aber wohl die Kenntnisse der Programmierer, die sie sich bei der Arbeit mit anderen Prozessoren angeeignet haben. Wer nur wenig Wissen von Maschinensprache besitzt, sollte mit dem Amiga diesen Sprung in die Maschine nicht wagen. Er ist mit C sicher besser bedient.

Als neuestes Werkzeug für die »unmenschlichste« Art der Programmierung bietet sich »Profimat« von Data Becker an. Dieses Programm ist ein kompaktes Entwicklungssystem, das dem Assemblerprogrammierer einen übersichtlichen Aufbau und leichte Bedienung garantiert. Der Lieferumfang des Paketes enthält neben einer Diskette ein über 140 Seiten umfassendes Handbuch, das durch einen kompakten Aufbau überzeugt. Es führt den Leser zu Beginn in den allgemeinen Umgang mit Assemblern ein und beschreibt anschließend die Arbeitsweise des Profimat. Nach Kenntnis- (oder Editieren) eines sogenannten Quellprogramms (eine Folge von Assemblerbefehlen, die als Quelle oder Eingabe für den Assembler dienen). Der Profimat-Editor wird bis auf kleine Ausnahmen durch Pull-Down-Menüs bedient und paßt sich somit dem Erscheinungsbild an, das sich auch schon bei anderen Amiga-Programmen bewährt hat. Die maximale Breite einer Zeile beträgt 127

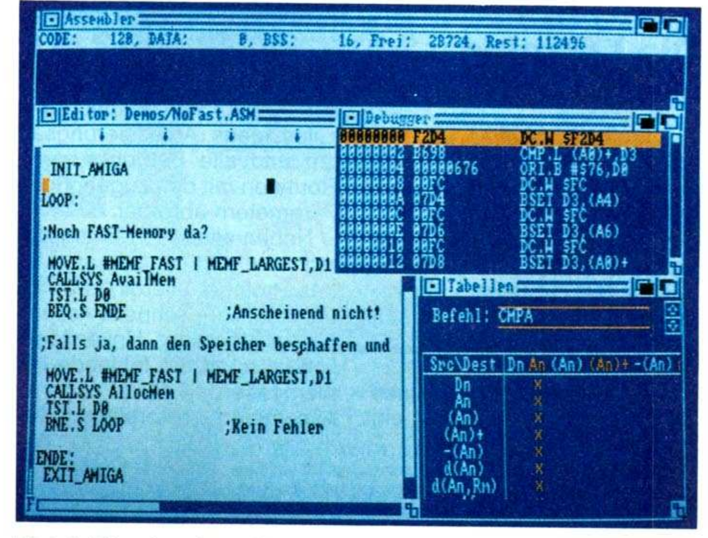


Bild 1. Die einzelnen Module des Profimat-Assemblers sind natürlich auch im Multitasking-Betrieb lauffähig

nahme der Installationshinweise und Anmerkungen zur Bedienung der Benutzeroberfläche ist der Leser bereits sehr schnell in der Lage, sein erstes Maschinenspracheprogramm einzugeben und auszuprobieren. Durch dieses erste Erfolgserlebnis motiviert, wird der engagierte Assembler-Fan das Buch gar nicht mehr aus der Hand legen.

Der Hauptteil der Dokumentation beinhaltet die Funktionsbeschreibung aller Menüpunkte und Befehle mit detaillierten Erklärungen und Beispielausdrucken. Zusätzliche Kapitel weisen in die Bedienung von Editor und Debugger ein. Eine Übersicht der verwendbaren Kommandos, der Assemblerbefehle und Fehlermeldungen rundet die Dokumentation ab.

Das eigentliche Programm besteht aus den vier Modulen Editor, Assembler, Debugger und Reassembler (Bild 1). Ein Editor ist im Prinzip eine »abgespeckte« Textverarbeitung. Einfache Funktionen (Tabelle 1) ermöglichen das Erstellen Zeichen. Dies reicht in der Regel vollkommen aus.

Der Programmierer kann nach der Fertigstellung des Quelltextes durch Wahl des entsprechenden Menüpunktes das Programm compilieren und starten. Der Profimat versteht den vollen Motorola-68000-Befehlssatz und erlaubt auch Alternativbefehle wie beispielsweise XOR und AND statt EOR und ANDI (Tabelle 2). Eine Reihe von Assemblerdirektiven (Pseudo-Opcodes) vereinfacht die Programmentwicklung (Tabelle 3).

So ist eine Include-Anweisung für das Einbinden von öfter verwendeten Variablendefinitionen oder Programmteilen vorhanden. Im Gegensatz zum Seka-Assembler von Kuma enthält der Profimat die Include-Dateien mit den Struktur- und Symboldefinitionen für das Amiga-Betriebssystem. Kontrollstrukturen wie REPEAT..UNTIL erleichtern die wiederholte Assemblierung einzelner Programmteile. Die bedingte Assemblierung mit IF,

Blockbefehle	
BA	markiere Blockanfang
BC	kopiere Block
BE	markiere Ende
BH	hebe Markierung auf
BK	lösche Block
BM	verschiebe Block
BP	füge Block ein
BV	zeige Block an
BX	schneide Block aus
Cursorsteuerung	
CL	Cursor nach links
CO	Cursor nach oben
CR	Cursor nach rechts
CU	Cursor nach unten
C(x,y)	setze Cursor auf Position x,y
CN	Cursor an den Anfang der nächsten Zeile
CV	Cursor an den Anfang der vorherigen Zeile
Löschen	
LA	lösche bis Zeilenanfang
LE	lösche bis Zeilenende
LL	lösche das Zeichen links vom Cursor
LR	lösche das Zeichen rechts vom Cursor
LZ	lösche Zeile
Tabulatorbefehle	
TL	Tabulator löschen
TS	Tabulator setzen
Sonstige Befehle	
Z Zeichenkette	Zeichenkette einfügen
ZH Zeichenkette	Zeichen hinter aktueller Zeile einfügen
ZV Zeichenkette	Zeichenkette vor momentaner Zeile einfügen

Tabelle 1. Der Editor des Profimat wird durch Menü- und einfach zu erlernende Tastatur-Kommandos gesteuert

IFD, IFND erlaubt eine Modifizierung des Maschinencodes durch Angabe bestimmter Parameter beim Start der Übersetzung. Damit kann zum Beispiel derselbe Quellcode auf verschiedene Computer angepaßt werden. Mit der Definition von Makros lassen sich spezielle Befehlsfolgen nur durch die Angabe des Makronamens in den Quelltext einfügen. Mit SLABEL kann der Programmierer sich eine Bibliothek mit Symbolen anlegen, die dann durch ILABEL in weiteren Programmen verfügbar gemacht werden können. Der Profimat-Assembler unterstützt außerdem lokale und redefinierbare Variablendefinitionen.

Motorola-Standard	Alternative
ADDA, ADDI	ADD
SUBA, SUBI	SUB
CMPA, CMPI	CMP
ANDI	AND
EORI, EOR	XORI, XOR
EOR	XOR
ORI	OR
MOVEA	MOVE

Tabelle 2. Der Profimat bietet bei den Befehlsnamen Alternativen zum Motorola-Standard, wie man ihn normalerweise gewohnt ist

Die Leistung überzeugt

Sämtliche Zeichen im Namen der Labels sind signifikant. Wer einmal längere Zeit einen Programmfehler gesucht und als Ursache letztlich eine Übereinstimmung der signifikanten Stellen unterschiedlicher Sprungmarken erkannt hat, weiß dies mit Sicherheit zu schätzen.

Beim Assemblieren eines Textes stellt der Profimat ein reichhaltiges Potential an Funktionen zur Verfügung, die sogar die Ausgabe einer übersichtlichen Cross-Reference-Tabelle ermöglichen. Sehr positiv sind auch die sogenannten Hilfstabellen zu bewerten. In ihnen sind sinnvollerweise alle 68000-Befehle mit ihren zulässigen Adressierungsarten und alle Betriebssystem-Routinen mit den zugehörigen Parametern abrufbar.

Neben all diesen Pluspunkten wird das Leistungskonto des Profimat allerdings durch die nicht sehr schnelle Assemblierung belastet. Unverständlich ist auch die fehlende Implementierung der in anderen Assemblern verbreiteten Befehle EVEN und CNOP. Dadurch wird die Übernahme von »fremden« Quelltexten unnötig erschwert. Das Handbuch gibt zwar Hinweise, wie man diese Befehle durch Makros selbst definieren kann, eine direkte Implementation wäre aber weniger umständlich gewesen.

Assemblerbefehl	Bedeutung
ALIGN.x	Sicherstellen, daß die nachfolgenden Befehle an geraden Adressen stehen
BSS	Beginn eines Block-Storage-Segments
CODE	Beginn eines Code-Segments
DATA	Beginn des Daten-Segments
DC.x	Daten-Definition von Daten beziehungsweise Werten, die als Konstanten im Speicher abgelegt werden
DS.x n[,Füllwert]	Reservierung eines Speicherbereichs der Länge »n« und dem Inhalt »Füllwert«
END	Obligatorischer Abschluß des Quelltextes
Name: Wert oder Name EQU Wert	Dem Symbol »Name« wird ein Wert zugewiesen
Name EQUR String	Einem Symbol wird eine Zeichenkette zugewiesen
IBYTES Datei[,Länge]	Daten, die auf dem externen Massenspeicher vorliegen, werden an der Stelle, an der dieser Befehl steht, eingefügt
IF; IFD; IFND	Bedingte Assemblierung.
INCLUDE Datei	Quelltextdatei einbinden
INPUT [Meldung,] Variable	Die »INPUT«-Funktion hat dieselbe Wirkung wie die normale Definition eines Speicherbereichs mit » = « oder »EQUR«, nur daß Sie den Wert während des Übersetzungsvorgangs festlegen können
LIST; NOLIST; PAGE	Druckerkommandos
MACRO	Makrodefinition
ORG Adresse	Startadresse des Programms
REPEAT; UNTIL	Wiederholte Assemblierung
SLABEL; ILABEL	Labelbibliotheksfunktionen

Tabelle 3. Die Pseudooperanden des Profimat überzeugen

Das Reassembler-Modul

Selbst wenn Sie kein Anfänger bei der Programmierung in Maschinensprache sind, müssen Sie den Begriff »Reassembler« nicht unbedingt kennen. Prinzipiell hat ein Reassembler dieselben Funktionen wie ein Disassembler: Er wandelt im Speicher befindliche Maschinencodes wieder in Assemblersprache um. Während die Ausgabe eines Disassemblers in der Regel nur für die Erzeugung von Assemblerlistings oder der Betrachtung kurzer Programmroutinen dient, können vom Reassembler erzeugte Dateien wieder als Eingabe für den Assembler genutzt werden. Zu diesem Zweck wird für jede Adresse, auf die in irgendeiner Weise von den Befehlen des zu reassemblierenden Bereichs zugegriffen wird, eine symbolische Bezeichnung (Label+1 für ungerade Adressen) generiert.

Der im Profimat integrierte Reassembler versucht, zwischen Daten und Programmroutinen zu unterscheiden. Er erkennt direkt angesprungene Unterroutinen, merkt sich das Ende dieser und erklärt alle nicht auf diese Weise durchlaufende Schritte als Daten. Indirekt angesprungene Befehle können so natürlich nicht als Programm erfaßt werden.

Der Debugger kann durch seinen geringen Leistungsumfang nicht überzeugen. So hilft er dem Anwender zwar, die (un)beliebten Guru-Meditation-Fehlermeldungen abzufangen, ein angeschlossener Drucker läßt sich damit jedoch nicht aktivieren. Der Einzelschrittmodus und die 68020-Einzelschritt-Emulation funktionieren zufriedenstellend. Ein Breakpoint kann allerdings nicht im ROM-Bereich positioniert werden.

Trotz der wenigen Schwachpunkte erhält sowohl der interessierte Einsteiger als auch der professionelle Programmierer mit dem Profimat ein Werkzeug, das mit so mancher Funktion konkurrenzlos ist. Der hohe Bedienungskomfort und die Funktionsvielfalt garantieren kurze Entwicklungszeiten für Assembler-Programme. Die zahlreichen Anpassungsmöglichkeiten erlauben bestens auf eine Aufgabe zugeschnittene Problemlösungen. Durch die Makrodefinitionen, die Unterstützung der 68010-Befehle und die Angabe von lokalen und redefinierbaren Variablen ist der Profimat auch für die Realisation von komplexen Programmstrukturen ideal geeignet. Der von vielen Data Becker-Produkten bekannte, niedrige Preis vereinfacht die Entscheidung für dieses Produkt, das sicher jeden Assembler-Profi überzeugen kann. (Markus Zietlow/pa)

AMIGA-WERTUNG

Software: Profimat

9,7 von 12

	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	■	■	■	■	■	■
Dokumentation	■	■	■	■		
Bedienung	■	■	■	■	■	
Erlernbarkeit	■	■	■	■		
Leistung	■	■	■	■	■	

Fazit: Mit seiner Funktionsvielfalt gehört der Profimat zu den besten Assembler-Paketen, die auf dem Markt erhältlich sind. Wegen des schwachen Debuggers ist das Programm allerdings weniger für die Untersuchung des Betriebssystems geeignet.

Positiv: Integrierter Reassembler; hoher Funktionsumfang des Assemblers; Include-Dateien im Lieferumfang

Negativ: Assemblierung relativ langsam; der wenig leistungsfähige Debugger unterstützt Breakpoints im ROM nicht

DATEN

Produkt: Profimat

Preis: 99 Mark

Hersteller: Data Becker

Anbieter: Gut sortierter Versand- und Fachhandel



Noch ein Briefkasten

AMIGA test

Mit »BBS-PC« sind Sie in der Lage, auf dem Amiga ohne große Programmierkenntnisse eine Mailbox, also einen elektronischen Briefkasten, zu realisieren.

Das Programm BBS-PC erschien schon vor einiger Zeit in den Regalen der Händler. Trotzdem wollen wir Ihnen dieses auf dem Amiga bisher einzigartige Programm vorstellen. Nicht zuletzt durch den anhaltenden Mailbox-Boom motiviert, kann man daran auch einige Vorurteile über sogenannte »Fertig-Mailboxen« widerlegen.

Als Käufer des BBS-PC erhalten Sie neben der eigentlichen Programmdiskette noch ein fast 200seitiges Handbuch. Trotz der ausführlichen Darstellung der einzelnen Funktionen wird man im Detail beim ausführlichen Studium einige Schwächen finden. Die Funktionsvielfalt beim BBS-PC ist derart erschlagend, daß dem Handbuch 100 Seiten mehr sicherlich keineswegs geschadet hätten. Das Programm ist voll multitaskingfähig und von der normalen Workbench aus zu starten. Die grundlegendsten Anforderungen an ein gutes Mailboxprogramm bestehen in der Datensicherheit, der schnellen Zugriffszeit auf die Daten, in der komfortablen Benutzer- und Sysopunterstützung sowie in den vielfältigen Funktionen wie Up-/Download und private Briefsektionen (Bild).

Vor allem in der schnellen Zugriffszeit tut sich BBS-PC hervor, da es seine Datensätze nicht in der üblichen sequentiellen Weise, sondern relational anlegt. Hier sticht auch die Verarbeitungsgeschwindigkeit des Programmes hervor. Bis zu 2400 Baud werden problemlos verarbeitet.

Relationale Daten

Programme können über die vier gängigsten Protokolle ausgetauscht werden; einer reinen Programmiererbox steht somit nichts im Wege.

Als Sysop können Sie natürlich jedem einzelnen Benutzer auch Prioritäten im Bereich von 0 bis 255 verleihen, damit eine effiziente Trennung von bestimmten Benutzergruppen realisiert werden kann.

16 Sektionen erscheint im ersten Moment recht viel, doch sind diese schnell aufgebraucht, da hier keine weiteren Untersektionen angelegt werden können.

Der jetzt erkannte Nachteil wird, leider nur etwas, durch die mitgelieferte Programmiersprache für Menüs und Titel

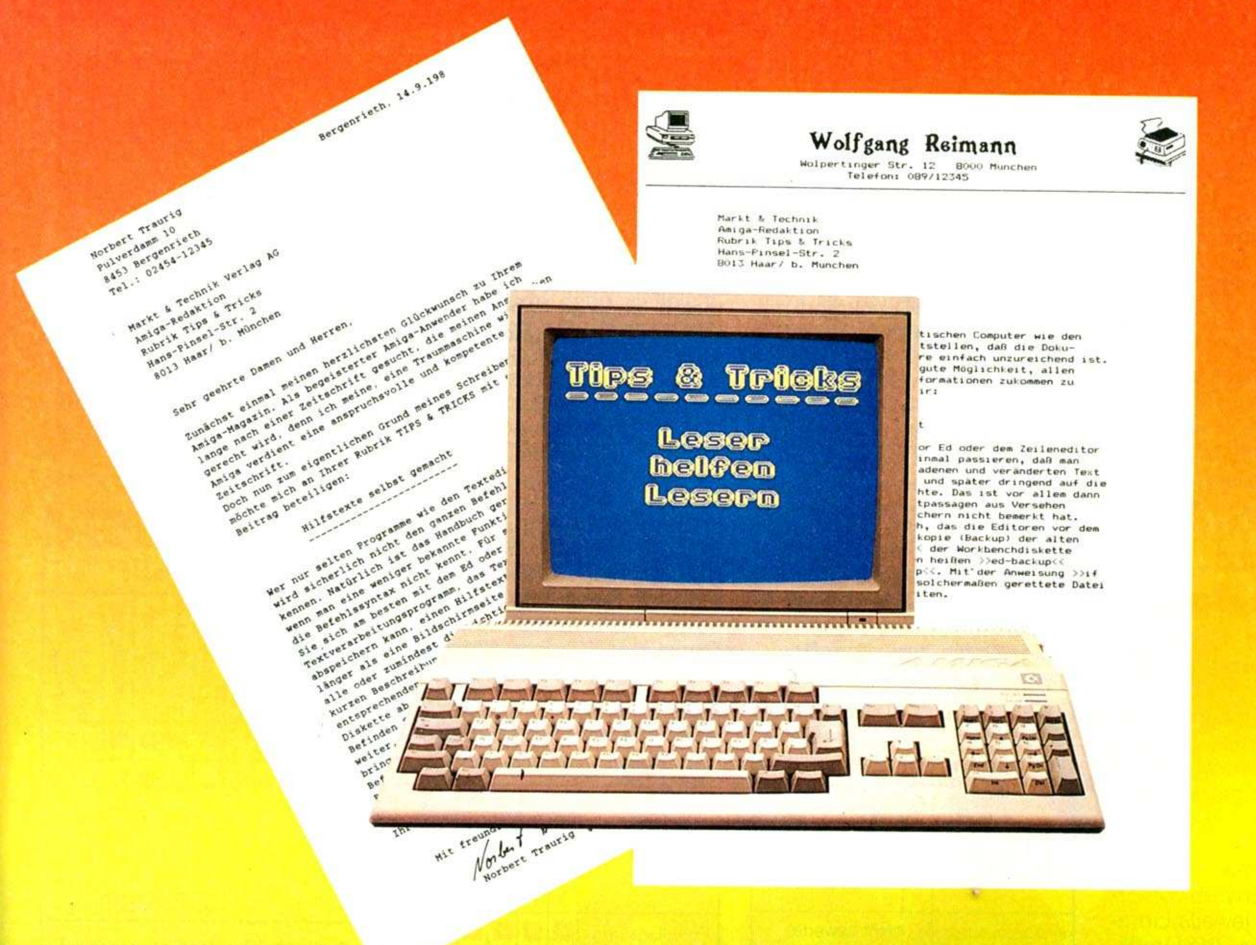
Fortsetzung Seite 133

```
+---------------------SYSOP MENU-+   +---------------------FILE MENU-+
! A: Add a new user              !   ! B: Browse files one at a time  !
! B: Enter BBS through back door !   ! C: Display file catalog        !
! C: Change a user's record      !   ! D: Download a file             !
! D: Delete a user               !   ! E: Exit back to main menu      !
! E: Exit back to main menu      !   ! H: Help with downloading files !
! G: Guest List (new users)      !   ! I: Information about files     !
! L: List current user file      !   ! K: Kill a file you uploaded    !
! M: Change modem defaults       !   ! L: Upload a file locally       !
! N: Display/edit section names  !   ! M: Mask out unwanted sections  !
! O: Purge multiple messages     !   ! N: Catalog of new files only   !
! P: Purge inactive users        !   ! R: Read a text file (formatted)!
! R: Reset member bulletin flags !   ! S: Search file descriptions    !
! S: Change user defaults        !   ! U: Upload a new file           !
! T: Change system defaults      !   ! *: Download direct from disk   !
! U: Globally edit user access   !   ! $: Upload direct to disk       !
! V: Edit node defaults          !   +--------------------------------+
! W: Edit terminal definitions   !
+--------------------------------+
```

Das Menü des Sysop- und des File-Moduls von BBS-PC

Gesucht:

Tips & Tricks zum Amiga



Sie sind ein begeisterter Amiga-Fan? Sie kennen Ihren Computer inzwischen so gut, daß die Begriffe »CLI«, »Amiga-DOS« oder »Workbench« für Sie keine Fremdwörter mehr darstellen? Dann beteiligen Sie sich doch an der Tips & Tricks-Rubrik »Ihres« neuen Magazins!

Die Tips & Tricks-Ecke im Amiga-Magazin ist eine reine »von-Lesern-für-Leser«-Rubrik. Das bedeutet, daß Sie durch Ihre Beiträge und durch Ihre guten Ideen anderen Lesern helfen können, mit dem neuen Computer besser zurechtzukommen. Sicherlich erinnern Sie sich noch an die Zeit, wo Sie begierig jede Zeile Information zum Amiga mit Heißhunger verschlangen. Gerade als Einsteiger ist man auf die Erfahrung und das Wissen von »Profis« angewiesen.

Sie haben eine neue Idee, wie man mit einem kleinen Amiga-DOS-Programm die Arbeit mit dem Computer vereinfachen kann? Einschicken!

Sie haben ein raffiniertes, kurzes Basic-, C- oder Assembler-Programm ausgeklügelt? Einschicken!

Sie haben einen Trick herausgefunden, wie man mit der einen oder anderen Software komfortabler arbeiten kann? Einschicken!

Sie haben Ihren Amiga durch eine kleine Bastelei aufgewertet? Einschicken! Jede gute Idee wird in »Ihrer« Rubrik veröffentlicht. Natürlich bekommen Sie dann von uns ein entsprechendes Honorar. Wichtig ist nur, daß Sie in einer kurzen Beschreibung den Grundgedanken Ihrer Idee zu Papier bringen. Wenn Sie ein Programm einschicken, sollten Sie nicht vergessen, eine Diskette mitzuschicken. Und sollte einmal ein Beitrag, aus welchen Gründen auch immer, nicht zur Veröffentlichung geeignet sein, bekommen Sie Ihre gesamten Unterlagen natürlich wieder zurückgesandt.

Schreiben Sie an:

Markt & Technik Verlag AG
Redaktion Amiga-Magazin,
Aktion Tips & Tricks
Hans-Pinsel-Str. 2
8013 Haar bei München

Eines sollten Sie nicht vergessen: Sie müssen kein Vollprofi sein, um bei dieser Aktion mitzumachen. Auch Tricks, die für Sie schon längst zur Routine geworden sind, könnten anderen Lesern eine wichtige Hilfe sein. Vor allem kurze und leicht durchschaubare Listings sind begehrt. Wir wünschen Ihnen viel Spaß beim Programmieren und Knobeln! (ub)

Fortsetzung von Seite 63

Disketten im Griff mit AmigaSort

— Beenden:

Dieser Menüpunkt schließt nach einer Sicherheitsabfrage alle Libraries und beendet das Programm. Taste: <HELP> + <ESC>.

• Ein weiteres wichtiges Menü ist »Drucken«, da hiermit die Liste auf dem Drucker ausgegeben werden kann. Das Programm verwendet den mit »Preferences« ausgesuchten Druckertreiber, den Sie natürlich vorher richtig eingestellt haben müssen.

— Name/Nummer:

Die aktuelle Liste wird mit Bezeichnungen und Nummern zweispaltig auf den angeschlossenen Drucker ausgegeben. Vorher muß die Workbench eingelegt werden, damit der Druckertreiber geladen werden kann. Taste: <HELP> + <F4>.

— Name/Diskette/Nummer:

Die aktuelle Liste wird mit Bezeichnungen, Diskettennamen und Nummern einspaltig auf den Drucker ausgegeben. Taste: <HELP> + <F6>.

— Alle Daten:

Die aktuelle Liste wird mit Bezeichnungen, Diskettennamen und Nummern sowie den Kommentaren auf den Drucker ausgegeben. Taste: <HELP> + <F7>.

Einen Beispielausdruck der drei verschiedenen Druckmodi mit unterschiedlichem Informationsgehalt von AmigaSort sehen Sie in Bild 2.

• Nun zu einem der wichtigsten Menüs von AmigaSort: »Suchen«. Damit Sie nicht immer die gesamte Liste durchblättern müssen, um einen Eintrag zu finden, können Sie die folgenden Menüpunkte anwählen.

— Namen:

Das Programm fordert zur Eingabe eines Suchbegriffs auf, der Bestandteil des gesuchten Eintrags sein muß. Auf Groß- und Kleinschreibung muß allerdings geachtet werden. AmigaSort durchsucht die Liste ab dem aktuellen Eintrag. Wird ein passender Eintrag gefunden, so wird er zum aktuellen gemacht, ansonsten meldet das Programm mit einem Pieps, daß kein passender Eintrag existiert. Taste: <HELP> + <F8>.

— Disketten:

Es gilt das gleiche wie oben, nur daß nicht nach Bezeichnungen, sondern nach Diskettennamen gesucht wird. Taste: <HELP> + <F9>.

— Nummern:

Das gleiche für Nummern. Hier wird allerdings nach dem Eintrag gesucht, der mit dem Suchbegriff identisch ist, und nicht wie oben nach einem, der den Suchbegriff irgendwo enthält. Taste: <HELP> + <F10>.

Natürlich ist es ohne Probleme möglich das Programm (siehe Listing) so abzuändern, daß zum Beispiel auch das Laufwerk »df2:« verwendet werden kann.

Für Anwender, die lieber mit den Tasten arbeiten, noch ein kleiner Tip: Zeichnen sie sich eine Schablone für die Funktionstasten, denn so haben Sie ständig die Belegung vor Augen.

Am schönsten und einfachsten ist jedoch eindeutig die Bedienung von AmigaSort mit der Maus. (Markus Säbel/rb)

Fortsetzung von Seite 116

Konkurrenz für Amiga-Basic

Man muß den Herstellern von True-Basic bescheinigen, daß ihnen mit ihrer Basic-Implementation auf dem Amiga eine guter Wurf gelungen ist, auch wenn typische Eigenschaften des Amiga, wie Animation und Unterbrechungsverarbeitung nicht berücksichtigt wurden. Dieser Mangel kann durch den Erwerb von Erweiterungen, die jeweils Libraries für Animation, 3D-Grafik, Stringbearbeitung und Sortieren enthalten, teilweise ausgeglichen werden. Aufgrund der hervorragenden Dokumentation kann True-Basic sowohl Anfängern als auch fortgeschrittenen Basic-Programmierern empfohlen werden. Ein etwas zwiespältigeres Bild liefert SAM-Basic. Im augenblicklichen Zustand erscheint es für Anfänger ungeeignet, während fortgeschrittene Programmierer, die sich nicht zur Sprache C durchringen können, in SAM-Basic das geeignete Entwicklungsinstrument finden. Der Bedienungskomfort ist zwar größer als bei C-Entwicklungssystemen, jedoch mit dem von Amiga-Basic und True-Basic nicht zu vergleichen. Insgesamt vermittelt SAM-Basic den Eindruck, als ob der Hersteller nicht den Mut gehabt hätte, sich in dem Umfang vom (veralteten) Konzept der Sprache Basic — wie sie etwa auf dem C 64 vorliegt — im gleichen Maße zu lösen wie Amiga-Basic und True-Basic. Dies ist um so bedauerlicher, als SAM-Basic sehr brauchbare Ideen enthält. Nach einer gründlichen Überarbeitung hat SAM-Basic gute Chancen, seine Mitbewerber um die Gunst des Käufers zu schlagen. (Jürgen Singer/pa)

AMIGA-WERTUNG

Software: Amiga-Basic

6,3 von 12

	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	nicht bewertet					
Dokumentation	■	■	■	■		
Bedienung	■	■	■	■		
Erlernbarkeit	nicht bewertet					
Leistung	■	■	■			

Fazit: Vom Leistungsumfang der am wenigsten befriedigende Interpreter.

Positiv: Interpreter wird kostenlos geliefert; Programm benötigt keine Zeilennummern

Negativ: Langsame Geschwindigkeit bei Ablauf und Editieren; beschränkte Erweiterungsmöglichkeiten; mangelnde Soundunterstützung

DATEN

Produkt: Amiga-Basic

Preis: kostenlos

Hersteller: Microsoft

Anbieter: Commodore; ist im Lieferumfang des Amiga enthalten

Vollständige Adressen entnehmen Sie bitte dem Anzeigenteil.

AMIGA-WERTUNG

Software: SAM-Basic

6,7 von 12

	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	■	■	■	■		
Dokumentation	■	■	■			
Bedienung	■	■	■			
Erlernbarkeit	nicht bewertet					
Leistung	■	■	■	■		

Fazit: Wegen der fehlenden Unterstützung durch die Dokumention eher für den fortgeschrittenen Programmierer geeignet

Positiv: Unterstützt den H.A.M.-Modus und Programmierung des Copper; verarbeitet IFF-Grafikdateien; komfortable Grafikanweisungen (inklusive 3D)

Negativ: Mäßige Dokumentation; Verwendung von Zeilennummern notwendig

DATEN

Produkt: SAM-Basic

Preis: 86,95 Pfund

Hersteller: PCC Ltd., 3 Mundells Court, Welwyn Garden City, Herts, AL7 1EN

Anbieter: für die Bundesrepublik noch nicht bekannt

AMIGA-WERTUNG

Software: True-Basic

7,7 von 12

	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	■	■	■	■		
Dokumentation	■	■	■	■	■	■
Bedienung	■	■	■	■	■	
Erlernbarkeit	nicht bewertet					
Leistung	■	■	■	■		

Fazit: Von der Konzeption her gut geeignet für Anwender, die auch mit dieser Basic-Version am PC arbeiten oder Zugang zu entsprechender Software haben. Durch die Implementation von Anweisungen zur Unterstützung der strukturierten Programmierung empfehlenswert für Einsteiger, die weniger an den Grafik- und Soundfähigkeiten des Amiga interessiert sind.

Positiv: Gute Dokumentation; Matrizenoperationen; schnelle Ablaufgeschwindigkeit; gute Unterstützung strukturierter Programmierung

Negativ: Keine Unterbrechungsverarbeitung, keine Spriteanimation; mäßige Soundunterstützung, englische Dokumentation

DATEN

Produkt: True Basic

Preis: 259 bis 549 Mark

Hersteller: True Corporation

Anbieter: Gut sortierter Fach- und Versandhandel

Vollständige Adressen entnehmen Sie bitte dem Anzeigenteil.

Fortsetzung von Seite 130

wettgemacht. So lassen sich jedenfalls eigene Menüs recht effektiv und künstlerisch interessant gestalten, wenn auch leider im gesamten BBS-PC Terminalemulationen wie etwa VT-52 oder ähnliche zur Einbringung von fetter und inverser Schrift fehlen. Wer jetzt denkt, auch eine vollkommen deutsche Box mit deutscher Menüführung und deutschen Umlauten zu konstruieren, sollte gewarnt sein. Es sind natürlich nicht alle Menüs veränderbar. So sind zum Beispiel die Befehlsübersichten und Befehlszeilen immer in Englisch, so daß höchstens eine gemischte Box mit großem Deutsch-Anteil entstehen könnte. Umlaute unterstützt BBS-PC in keinster Weise, vom PAL-Modus einmal ganz zu schweigen.

Trotzdem sollten Sie dieser, Basic-ähnlichen, Programmiersprache nicht den Rücken kehren, da dort noch einige Funktionen der Mailbox ungenutzt brach liegen. Haben Sie das Programm geschrieben, kann es, wie jeder andere Sourcecode auch, nicht sofort ausgeführt, sondern muß erst compiliert werden. Das geht mit Hilfe des mitgelieferten Compilers recht zügig.

Ein weiterer wichtiger Punkt ist die Leistungsfähigkeit des Editors zur Eingabe von Briefen. Hier glänzt BBS-PC wieder in allen Funktionen, die gut durchdacht, aber durch das bei uns in Deutschland unübliche Konzept anfangs verwirrend sind. Einmal verstanden macht es aber großen Spaß, mit diesem Editor zu arbeiten, der sogar Text-Uploads einwandfrei — ohne das Verschlucken von Zeichen — verkraftet.

Amerikanische Mailboxfreaks werden sicherlich wesentlich weniger Probleme mit dem Editor haben, da dieser, wie auch das gesamte Programm in seinem Aufbau, dem amerikanischen Compuserve-Net angeglichen ist.

Nach einiger Zeit kommen Sie als Sysop dann vielleicht auf die Idee, das Unwichtige aus solchen überlangen Briefen herauszustreichen und damit den Informationsgehalt zu erhöhen. Doch bald stoßen Sie auch hier wieder auf einen gravierenden Mißstand: Es ist nicht möglich, einmal abgeschickte Briefe zu editieren. Zwar können Sie diese sperren, nur bestimmten Personen zugänglich machen oder auch ganz löschen, aber das Editieren ist unmöglich.

Ein weiteres Problem, das durch die Verwendung relationaler Dateien entsteht, ist die Wiederherstellung der ursprünglichen Form nach einem Computerabsturz. Eine kleine Hilfe in solchen Fällen ist das mitgelieferte Fix-Programm, das, soweit irgend möglich, die Datensätze wieder lesbar macht.

Alle Pfade und Namen zu relevanten Dateien oder Verzeichnissen können selbstverständlich frei gewählt werden, so daß sich eine Hard-Disk für diese Zwecke geradezu anbietet. Aber auch mit viel Speicher und einer resetfesten RAM-Disk lassen sich Dateien schnell abrufen.

Damit bei Ihnen kein falscher Eindruck entsteht: Hier konnte nur ein Bruchteil der Funktionen des BBS-PC angesprochen werden. Es stehen insgesamt wesentlich mehr Möglichkeiten zur Verfügung, die Mailbox an die eigenen Wünsche anzupassen.

(Ottmar Röhrig/jk)

AMIGA-WERTUNG

Software: BBS-PC

6,0 von 12

	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	■	■	■			
Dokumentation	■	■	■	■	■	
Bedienung	■					
Erlernbarkeit	■	■	■			
Leistung	■	■	■	■		

Fazit: BBS-PC ist für jeden angehenden Sysop interessant, da es viele Funktionen zu einem fairen Preis bietet. Trotzdem ist teilweise viel Tüftelei gefragt, bis Sie zum gewünschten Ergebnis kommen.

Positiv: Eingebaute Hilfsprogramme; schneller Datenzugriff; freier Menüaufbau; multitaskingfähig.

Negativ: Datensicherheit nicht immer gewährleistet; in keiner Weise Intuition-orientiert; begrenzte Brieflänge; kein Editieren von Briefen.

DATEN

Produkt: BBS-PC

Preis: 193 Mark

Hersteller: Micro-Systems-Software

Anbieter: Softwareland, Franklinstr. 27, CH-8050 Zürich, Tel. 0041-1-3115959

Der Filmrecorder

AMIGA test

Wie von einer Filmspule kann man jetzt IFF-Bilder auf dem Amiga abspielen: Einfache Animationen mit »Page Flipper«.

Der »Page Flipper« ist ein Tool, mit dessen Hilfe nichts anderes gemacht wird, als Einzelbilder hintereinander abzuspielen. So kann man, wie bei einem echten Zeichentrickfilm, jede Einzelphase extra zeichnen und ein kleines Meisterwerk daraus machen. Ein einfaches Bilderwechseln ist das aber nicht, denn der Page Flipper versteht sich auf mehr als nur stures Umschalten zwischen Grafiken. Das sieht man auch am dicken 64seitigen Handbuch, das beileibe nicht den Eindruck eines undurchdachten Programmes macht.

So kann das Arbeiten mit dem Page Flipper gleich von Anfang an mit Hilfe des Handbuchs erlernt werden. Dabei erfährt man nicht nur, wie mit dem Produkt richtig umzugehen ist, sondern auch Wissenswertes über andere Grafikprogramme, deren Bilder man verwerten kann. Warum zum Beispiel Deluxe-Paint II nur 672 x 444 Pixel verwendet, und Aegis-Produkte 480 horizontale Linien im HiRes-Overscan-Modus benutzen. Natürlich werden einem auch die Auswirkungen auf die Programmierung des Page Flippers erklärt, der immer den neuesten Grafikprodukten angepaßt wird.

Page Flipper arbeitet inzwischen mit Grafiken aus Videoscape, Deluxe-Paint II, Aegis Images, Sculpt 3D, Silver, Digi View und Digi Paint. Eine Anpassung an den neuen RGB4-Modus (Prism Plus, ohne das alte HAM-Problem) und andere neue Grafikprodukte, zum Beispiel mit HAM-Overscan, ist geplant. Page Flipper selbst arbeitet mit allen Grafikauflösungen und Grafikmodi außer Extra-Halfbright und geht bis zu PAL-Overscan. Alle Grafiken einer Filmsequenz müssen allerdings immer in derselben Grafikauflösung und dem gleichen Modus sein und die Bitplane-Zahl eines jeden Bildes muß identisch sein.

Damit sind wir schon beim Thema: das Erstellen von Filmsequenzen. Nachdem sich das Programm automatisch an die Auflösung des ersten Bildes einer Sequenz anpaßt (wenn nicht ausdrücklich im Script-File die Auflösung genannt wurde), spielt es die Grafiken nach der im Script-File abgelegten Befehlsreihenfolge ab. Befehle gibt es nicht nur zum Anzeigen von Bildern, sondern auch für Pausen, für das Wechseln der Auflösung vor dem Anfang einer neuen Sequenz, oder zum Beispiel für das Timing. Ein Film, der die Bilder zu schnell abspielt, ist nämlich nur in Sonderfällen erwünscht. Es macht schließlich einen Unterschied, ob der gute Donald der geliebten Daisy einen innigen Kuß oder einen typischen Schnellkuß gibt (Bild). Dabei kann man eine Pausenlänge für eine ganze Sequenz generieren, aber auch die Pausen zwischen jedem einzelnen Bild differenziert angeben.



Der »Page Flipper« macht aus IFF-Bildern Filme und spielt sie in beliebiger Geschwindigkeit ab

Schon mit vier bis fünf Bildern kann eine eindrucksvolle Animation entstehen, wenn man sie in der richtigen Reihenfolge und mit den entsprechenden Pausen abspielt.

Neben den Sequenzfunktionen für den Film gibt es noch verschiedene Kontrollfunktionen, die man sowohl über Menü als auch per Script-Befehlen ausüben kann. »Delay« regelt die Geschwindigkeit des gesamten Films, so daß nicht jede Pausenfunktion extra angepaßt werden muß. »Button« regelt den Einfluß der Maus: Entweder wird im Einzelschrittmodus Bild für Bild mit der Maus umgeschaltet, oder Sie können während des normalen Abspielens den Film rückwärts laufen lassen. Im Einzelschrittmodus hat man so aus seinem Amiga auch gleich einen kleinen Diaprojektor gemacht. Die Filmrolle wird quasi als Endlosrolle am Ende zusammengeklebt, sobald sie die »Continuous«-Option wählen. Letzter Punkt der Kontrollfunktionen: »Motion«, der das »Double Buffering« an- oder ausschaltet. Mit besonders vielen Einzelbildern in der Sekunde (in der Regel bei mehr als 20) wird nämlich das Bild flimmerig, wenn es in derselben Grafikseite aufgebaut wird. Um das Bildflattern zu vermeiden, baut sich in »DBUF-Motion« das nächste Bild schon im Hintergrund auf, während das erste noch sichtbar ist. Erst nach dem Kopieren des Bildes in die Hintergrundseite wird umgeschaltet.

Bis zu 8,5 MByte, also den gesamten adressierbaren Speicherraum des Amiga, kann man mit Bildern vollstopfen. Ein gewaltiges Potential, das sich dem Filmhobbyisten da auftut. Und der Vorteil: Man braucht keinerlei Funktionsmechanismen irgendwelcher spezieller Animationsprogramme zu kennen — einfach nur Bild für Bild malen oder von Programmen konstruieren lassen. Ein idealer Einstieg in den Computerfilm also. Mit einem verständlich und ausführlich geschriebenen (englischen) Handbuch beweist der Page Flipper auch, daß er gerade für Einsteiger ideal geeignet ist.

Hat man allerdings nur 512 KByte Speicher, ist es wahrlich eine kleine Filmspule, die man bearbeiten kann: Sehr viele Bilder bekommt man in der Regel nicht in das Chip-Memory. Doch auch damit lassen sich schon beachtliche Erfolge erzielen.

Ein Bonbon zum Schluß: Wer die beiliegende Karte ausfüllt und an den Hersteller sendet, bekommt automatisch die neueste Version des Programms, das laut Autoren immer dem neuesten Stand von Grafikprogrammen angepaßt wird. Ein Gutschein für einen Preisnachlaß auf das Nachfolgeprogramm »Page Flipper Plus«, das eine professionellere Zielgruppe ansprechen soll (mehr als 60 Bilder pro Sekunde mit neuen Features), ist als kleine Beigabe für den treuen Kunden enthalten.

(M. Kohlen/jk)

AMIGA-WERTUNG

Software: Page Flipper

8,6 von 12

	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	■	■	■	■	■	
Dokumentation	■	■	■	■	■	■
Bedienung	■	■	■	■	■	
Erlernbarkeit	■	■	■	■	■	
Leistung	■	■	■	■		

Fazit: Page Flipper ist nicht gerade das, was man ein professionelles Werkzeug für den Videoeinsatz nennt, aber eignet sich ideal für den Einstieg und deckt eine Lücke der bisherigen Animationsprogramme (die Einzelbildschaltung) ab.

Positiv: Gutes Handbuch als ideale Einführung; einfache Bedienung

Negativ: Alle Bilder einer Sequenz müssen gleiche Auflösung haben; Scriptfile nur mit ASCII-Editoren zu ändern.

DATEN

Produkt: Page Flipper

Preis: 98 Mark

Hersteller: Mindware International

Anbieter: Atlantis, Dunantstr. 53, 5030 Hürth, Tel. 02233/41081

DTM, Poststr. 25, 6200 Wiesbaden, Tel. 06121/560084

Frühtau auf Computerbergen

AMIGA test

Wenn Sie von Landschaftsformationen aus dem Computer fasziniert sind, sollten Sie sich den »Landscape-Designer« anschauen. Er bietet jetzt eine preiswerte Alternative für Grafikbegeisterte.

Den frischen Sauerstoff in den luftigen Regionen ersetzt Landscape-Designer zwar nicht, aber Berge und Täler auf Mausklick mit dem Computer zu kreieren, ist die Domäne des Programmes. Daß das Programm aus bayerischen Landen stammt, ist nicht weiter verwunderlich. Was kann der Landscape-Designer und für welche Zielgruppe ist er gedacht? Das Programm bietet Ihnen in erster Linie eine Art Punkt-Editor zum Erstellen von Landschaften (Bild 1). Der Begriff Landschaften bezieht sich hierbei allerdings wirklich nur auf die Höhe eines Punktes. Bäume, Straßen oder Trassen können nicht erzeugt werden. Zur Bestimmung des Aussehens der Landschaft verwendet Landscape-Designer einen zweidimensionalen Editor, der die Ansicht des Gebietes von oben zeigt. Die Höhe der einzelnen Punkte wird dabei durch ihre Farbe kenntlich gemacht.

Diese Farbe richtet sich nach einer angewählten (und vom Benutzer editierbaren) Farbpalette, die in allen Abstufungen rechts auf dem Menü-Bildschirm dargestellt wird (Bild 2). Dies hat den Nachteil, daß oft die Höhe der Punkte nicht einwandfrei zu unterscheiden ist, da die Farbabstufungen beim Amiga bekanntlich sehr gering sein können.

Doch weiter mit dem Editor. Sie können einzelne Punkte setzen oder ganze, vordefinierte Berge beziehungsweise Täler auf dem Boden plazieren. Dabei werden diese Formen natürlich auch aus einzelnen Punkten zusammengestellt. Es stehen Ihnen sieben Formen für Hügel oder Täler und eine Limiterfunktion zur Erstellung von Plateaus sowie die schon erwähnte Punkt-Funktion zur Verfügung, um Ihre Landschaft zu gestalten.

Ist eine Form angewählt, ist es möglich, die Höhe dieses Berges oder der Rampe (dazu gleich mehr) durch Einstellen des Höhenreglers zu verändern. Diese Höhe kann zwischen —100 und 100 vorgewählt werden, wobei die Höhe null den Erdboden darstellt und negative Werte ein Tal kennzeichnen.

Um nun möglichst realistische Gebirge erstellen zu können, sind die sieben Formen wie folgt definiert: Berg mit runder Grundfläche, Berg mit rechteckiger Grundfläche, Pyramide, Quader, schiefe Ebenen in der X-Achse, schiefe Ebenen in der Y-Achse und Zufallsstrukturen. Weiterhin steht eine Art Lösch-Funktion zur Verfügung, die alle eingegrenzten Punkte auf ein bestimmtes Niveau zurücksetzt; wird hier Null angegeben, lassen sich auf diese Weise falsch gesetzte Punkte und Punkt-Gebilde wieder löschen. Die letzte Funktion ist für die nachträgliche Änderung der Höhe von Bergen oder der Tiefe von Tälern. Dabei wird aber nicht die wirkliche Höhe des Objektes modifiziert, also vergrößert oder verkleinert, sondern bei der eingestellten Höhe beziehungsweise Tiefe ein Schnitt gemacht und der Berg oder ähnliches an dieser Stelle abgeschnitten. Damit entstehen auf wirkungsvolle Weise Plateaus, die dann weiter mit Objekten besetzt werden können.

Ihr Programm im Amiga-Magazin



Haben Sie bereits ein interessantes Programm für den Amiga geschrieben? Dann sollten Sie sich vielleicht überlegen, ob das von Ihnen entworfene Programm nicht für die Allgemeinheit der Amiga-Besitzer nützlich ist.

Nicht nur neue Spiele werden durch neue und leistungsfähige Computer wie den Amiga zu faszinierenden Meilensteinen in der Heimcomputer-Geschichte. Durch die steigende Speicherkapazität und die immer professionelleren Betriebssysteme können auch immer mehr höhere Programmiersprachen eingesetzt werden. Da die einzelnen Systeme heute immer komplexer aufgebaut sind, werden sogar Betriebssysteme in einer höheren Sprache erstellt. Auch hier ist der Amiga wieder ein sehr gutes Beispiel. Seine Seele, das Betriebssystem, wurde mit der modernen Sprache C programmiert.

Da für den Amiga in der Zwischenzeit ein bunt gemischtes Sammelsurium an verschiedensten Sprachen zur Verfügung steht, von Fortran bis Prolog, versorgen wir Sie natürlich mit brandheißen Informationen zu diesen Sprachen. Dabei wollen wir selbstverständlich nicht an unseren Lesern »vorbei programmieren«. Vielmehr haben Sie hier die Möglichkeit, Ihr Amiga-Magazin als Leser mitzugestalten. Wir suchen vor allem kurze Programme mit großer Wirkung, also Utilities oder wirkungsvolle Routinen, die jeder Leser und Programmierer sehr leicht für sich auswerten kann. Besonders nützlich ist auch eine genaue Dokumentierung des Quellcodes. Dadurch erschließt sich die von Ihnen verwendete Programmtechnik auch allen anderen Amiga-Anwendern. Die Programme werden leicht nachvollziehbar, und bestimmte Teile können so auf einfache Weise in eigene Programme eingebaut werden. So steigert sich ganz allmählich auch die Qualität der veröffentlichten Programme. Natürlich wird Ihr Beitrag bei Veröffentlichung entsprechend honoriert. Welche Sprache oder welchen Compiler Sie dabei verwenden, ist vollkommen egal. Ob Sie nun das bereits vorhandene Amiga-Basic benutzen oder sich mit Maschinensprache und C herumschlagen, wir werden Ihre Einsendungen bewerten und, wenn für gut befunden, in einer der folgenden Ausgaben veröffentlichen. Wie gesagt, sie können jede Sprache verwenden, die für den Amiga erhältlich ist.

Wie gehen Sie bei der Einsendung vor? Ganz einfach, schicken Sie uns Ihr Programm mit genauer Beschreibung und Diskette, sowie der Angabe der verwendeten Sprache an folgende Adresse:

Markt & Technik Verlag AG
Amiga-Redaktion
Stichwort:
Amiga-Programmierung
Hans-Pinsel-Str. 2

8013 Haar bei München

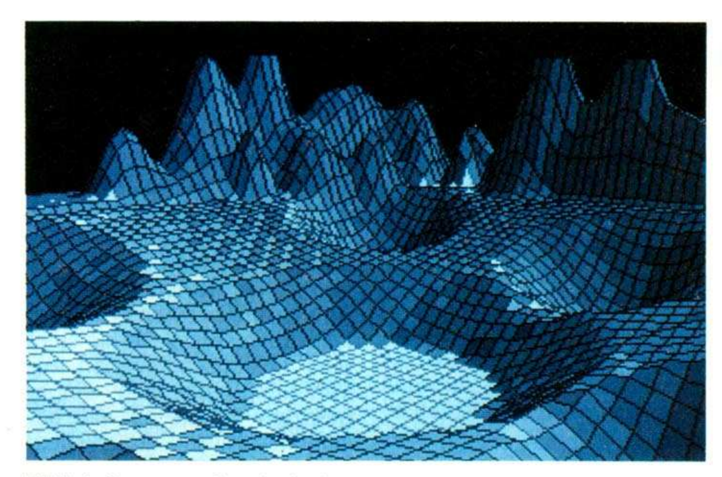


Bild 1. Computerlandschaften mit beliebiger Struktur und verschiedenen Farben können entworfen werden

Die Größe der Landschaft ist leider nicht frei wählbar, sondern auf das Bildschirmformat festgelegt. Einige Gadgets erlauben es Ihnen jedoch, die Auflösung dieses Gebietes zu wählen. Es stehen Grundraster von 18 x 18, 36 x 36 und 72 x 72 Feldern zur Verfügung. Um so größer Sie die Auflösung wählen, um so größer ist natürlich auch der Rechenaufwand und die Berechnungszeit schnellt in die Höhe. Aber auch das Zeichnen der Landschaft (berechnen und zeichnen sind zwei unmittelbar aufeinanderfolgende aber doch getrennte Schritte) dauert natürlich länger als bei niedrigeren Auflösungen. Doch auch der Nutzen dieser Wahl soll nicht verschwiegen werden. So werden die erstellten Landschaften natürlich wesentlich »naturgetreuer«, und es fehlen die scharfen Kanten, die durch die Verwendung von Quadraten oder Gittern zum Bildschirmaufbau verursacht werden (Bild 3). Alles in allem sollten fertige Landschaften immer im Modus mit der höchsten Auflösung berechnet werden, da der vom eigentlichen Bild benötigte Speicherplatz natürlich gleich bleibt. Erfreulich ist in diesem Punkt, daß auch während der Arbeit an einem Projekt dessen Auflösung verändert werden kann. Dabei ist es jedoch verständlich, daß beim



Bild 2. Der Menü-Bildschirm des Landscape-Designers mit der gut durchdachten Farbpalette (rechts)

Transferieren in niedrigere Auflösungen gewisse Details verlorengehen. Die Umrechnung dauert zwar manchmal einige Zeit (niemals mehr als 2 bis 3 Minuten), aber es funktioniert — und das ist das Wichtigste. Um die Rechenzeit für Vorentwürfe zu verringern, stehen Ihnen auch mehrere verschiedene Zeichenmodi zur Verfügung. Diese arbeiten entweder ohne oder mit Hidden-Line-Algorithmus oder sparen über bestimmte Einfärbung des Gitters Rechenzeit.

Zeitprobleme entstehen leider beim Neuaufbau jeder Zeichnung und beim Laden von Objekten zur Editierung oder Berechnung. Hier sind Wartezeiten an der Tagesordnung — die Probleme liegen hier sicherlich nicht bei der langsamen Verarbeitungsgeschwindigkeit des Amiga.

Weiterhin ist es möglich, eine Lichtquelle zu positionieren, die dann in die Berechnung der Helligkeit der einzelnen Flächen mit einbezogen wird. Bei der Berechnung werden leider keine Schatten von über dem Betrachter liegenden Bergen oder ähnlichem erkannt. Trotzdem lassen sich durch die zusätzliche Shade-Funktion noch etwas realistischere Bilder erzeugen, als dies mit einer einfachen Lichtquelle machbar wäre. Es besteht auch die Möglichkeit, ein Objekt aus der Berechnung der Farbübergänge auszunehmen und nur in einer Farbe erscheinen zu lassen. Dabei wird aber schnell der mühsam geschaffene, realitätsnahe Eindruck zerstört.

Neben dem Betrachtungswinkel sind Sie ebenfalls in der Lage die Größe des gesamten Gebietes zu verändern, womit Zoom-Effekte erzeugt werden können.

Beim Experimentieren mit den einzelnen Farbpaletten, die alle nachträglich geändert werden können, stellt sich schnell ein Verlauf von Weiß nach Schwarz für Tiefen beziehungsweise Höhen als die zweckmäßigste Darstellungsart heraus. Teilweise kommt es aber bei der Verwendung von Original-Paletten dazu, daß die Gadgets und Menüs nicht mehr lesbar sind und Sie für eine Zeit »blind« arbeiten müssen. Erfreulicherweise arbeitet Landscape-Designer wahlweise im NTSC- oder im PAL-Modus, wobei bei fehlendem Speicher die Umschaltung in den PAL-Modus leider zu Abstürzen führen kann. Es ist allerdings schade, daß der schon bei so vielen Programmen Einzug gehaltene Overscan-Modus nicht unterstützt wird. Im 20seitigen, teilweise etwas zu locker geschriebenen, deutschen Handbuch werden alle Funktionen gut erklärt. Darin wird der Benutzer aber nicht gerade mit Fakten erschlagen. Inwieweit das Programm für diesen Preis interessant ist, muß jeder selbst entscheiden. Mal- oder Animationsprogramme sind die Werkzeuge, die mit den durch Landscape-Designer entstandenen Bildern etwas anfangen können. Da bei anderen Profiprogrammen ähnliche Editoren bereits enthalten sind, muß jeder für sich selbst abwägen, inwieweit er den Landscape-Designer wirklich benötigt.

(Ottmar Röhrig/jk)



AMIGA-WERTUNG

Software: Landscape-Designer

7,2 von 12	ungenügend	mangelhaft	ausreichend	befriedigend	gut	sehr gut
Preis/Leistung	■	■	■	■	■	
Dokumentation	■	■	■			
Bedienung	■	■	■	■		
Erlernbarkeit	■	■	■			
Leistung	■	■	■	■		

Fazit: Der Landscape-Designer ist eine interessante 3D-Grafiksoftware, die jedoch viele kreative Ideen fordert. Das Programm allein kann zur Erstellung von einfachen Computerlandschaften benutzt werden. In Verbindung mit guten weiteren Grafikprogrammen sind jedoch mit einigem Aufwand viele Vorstellungen realisierbar.

Positiv: PAL/NTSC-Unterstützung; leichte Bedienbarkeit; relativ zügige Berechnung.

Negativ: Zu langsame Laderoutinen; kein Overscan; großer Speicherplatzbedarf; zu einfaches Handbuch.

DATEN

Produkt: Landscape Designer

Preis: 69 Mark

Hersteller/Anbieter: C. Obermaier, Karl-Marx-Str. 1, 8000 München 71, Tel. 089/ 791 44 39

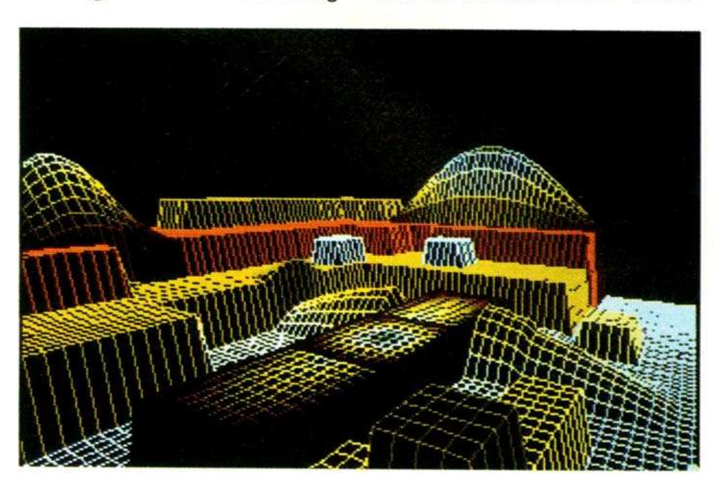


Bild 3. Mit abgerundeten Ecken und Kanten wirken die Grafiken wesentlich naturgetreuer

PROGRAMM-SERVICE

IFF-Grafik-Show für Anspruchsvolle

IFF-Diashow: Als Resultat des in Ausgabe 6-7/87 gestarteten Malwettbewerbs präsentieren wir Ihnen zwei randvoll bespielte Disketten, die mit den besten Bildern des Wettbewerbs (siehe Amiga-Magazin 10/87, Seite 129) bespielt sind. Die Bilder können entweder mit dem enthaltenen Diashow-Programm angesehen oder mit jedem gängigen IFF-Malprogramm geladen werden. Lassen Sie sich überraschen, welche hochwertigen und vor allem sehenswerten Grafiken unsere begabten Leser geschaffen haben.
Zwei Disketten für den Amiga

Bestell-Nr.: 49705 **DM 29,90*** sFr 24,90*/öS 299,-*

Berechnung realistischer Bilder mit 3D-Effekt

Raytracing: Mit dem Programm des Monats können Sie fantastische Bilder berechnen lassen, wobei der Strahlengang des Lichts berücksichtigt wird. **Termin:** Sie vergessen nie wieder Termine mit diesem Programm, das die Daten der nächsten 15 Tage automatisch anzeigt. **Chain Reaction:** Ein spannendes Spiel, das viel Taktik erfordert und praktisch nur auf dem Computer realisierbar ist. Lösen Sie eine Kettenreaktion aus! **EHB-Demo:** Wie man 64 Farben im Lores-Modus benutzen kann, zeigt dieses C-Programm, das den Extra-Half-Brite-Modus verwendet. **Copper:** Auch von Basic aus kann der Copper für erstaunliche Effekte eingesetzt werden. Die Verwendung der nötigen Bibliotheken können Sie dabei erlernen. **Cursor Cycle:** Diese Utility macht Schluß mit dem unscheinbaren Cursor, indem es dessen Farbe ständig ändert. **MyCLI:** Eine unentbehrliche Hilfe für alle, die CLI-Fenster mit besonderer Voreinstellung benötigen. **MEd:** Das Schreiben von C-Programmen für Pull-down-Menüs verkürzt sich auf wenige Minuten. Mit einem weiteren Teil können Sie dann CLI-Befehle im Pull-down-Menü verwenden.
1 Diskette für Amiga

Bestell-Nr.: 48704 **DM 29,90*** sFr 24,90*/öS 299,-*

Super-Malprogramme

Rainbow-Drawer: Dieses Programm des Monats bietet leistungsfähige Befehle und Funktionen, wie sie von professionellen Programmen bekannt sind: bis zu 32 Farben, alle Auflösungen, viele Befehle zum Zeichnen sowie FILL mit Mustern, BOW und anderem. **Turtle:** Mit dieser Befehlserweiterung verfügen Sie über die Grafikbefehle, die bei Logo bekannt und beliebt sind. **Fractals:** Dreidimensionale, realistische Gebirge mit Schattierung erzeugt dieses Programm. **Clouds:** Genauso wirklichkeitsnah wie die Gebirge, aber noch erstaunlicher, sind die Wolken, die Sie mit Clouds generieren. **Apfelmännchen:** Hiermit erzeugen Sie schöne Grafiken aus der beliebten Mandelbrot-Ebene. **Kudiplo:** Ein gutes, unverzichtbares Werkzeug für die Kurvendiskussion stellt »Kudiplo« dar. **Senso:** Testen Sie mit dieser Computer-Adaption des bekannten Spiels Ihr Gedächtnis! **Division:** Bis zu 32000 Nachkommastellen können durch dieses Programm berechnet werden. **Alert:** Alarme, zum Beispiel die bekannten Guru-Meditations, können Sie nun selbst programmieren. Das Programm ist in erster Linie für C-Programmierer aufschlußreich. **Border:** Lassen Sie den Fensterrahmen des CLI-Fensters einfach verschwinden! **SCD:** Mit diesem Utility können Sie den Pfadnamen in der Titelleiste des Fensters anzeigen.
1 Diskette für Amiga

Bestell-Nr.: 48703 **DM 29,90*** sFr 24,90*/öS 299,-*

Disk-Operation vereinfacht

ExDir: Mit dem Programm des Monats der Ausgabe 8/9 sind alle Schwierigkeiten mit den Subdirectories endlich vorbei. **Hardcopy:** Diese nützliche Routine gestattet es, Teile des Bildschirms auszudrucken. **Bio:** Um Ihre körperliche, seelische und psychische Verfassung zu erfahren, können Sie dieses fantastische Biorhythmusprogramm einsetzen. **Mask:** Dieses Utility hilft Ihnen beim Herstellen komplexer Eingabemasken. **Freeman:** Den momentan verfügbaren Speicherplatz erfahren Sie mit dieser Routine im CLI-Window. **Vokabel:** Zum Lernen von Vokabeln ist dieses Programm ein sehr gutes Werkzeug mit viel Komfort. **SimpleSprite:** Wie man Sprites auf dem Amiga programmiert, sehen Sie in diesem Demoprogramm. **Vier gewinnt:** Wenn Sie gern spielen und strategisches Denken trainieren wollen, ist Vier gewinnt genau das Richtige für Sie. Weiterhin finden Sie auf der Diskette ein nützliches Tool zur Veränderung des CLI-Windows (InitCLI).
1 Diskette für Amiga

Bestell-Nr.: 48702 **DM 29,90*** sFr 24,90*/öS 299,-*

Leckerbissen für C-Programmierer

Netzgrafik: Wie Sie auf dem Amiga dreidimensionale Netzgrafiken erstellen, zeigt dieses Programm in Amiga-Basic. **Disketten-Monitor:** Dieser Monitor, geschrieben in Amiga-Basic, bietet viele Möglichkeiten bei der Diskettenmanipulation. **Menü-Routinen:** Benutzen Sie die Betriebssystemroutinen von C zum Erzeugen von Pull-down-Menüs. **Polydraw:** Wie Sie die Grafik-Bibliothek in Amiga-Basic sinnvoll nutzen, zeigt dieses Programm. **Font-Designer:** Amiga-Basic-Programm, das Ihnen ermöglicht, eigene Zeichensätze herzustellen. **FTYPE:** In C geschriebene Druckroutine mit viel Komfort **FCat:** Schnelle Directory-Routine. **Objekt-Editor:** Ein Sprite- und BOB-Editor mit komfortabler Maussteuerung.
1. Diskette für Amiga

Bestell-Nr.: 48701 **DM 29,90*** sFr 24,90*/öS 299,-*

Super-Utilities für Grafik, Sound und Disk

Brush-Converter: Ein Programm, das Deluxe-Paint-Brushes in Amiga-Basic-Objekte umsetzt. **Gesellschaftsspiel:** Punktesammeln durch das richtige Beantworten vieler Fragen aus den unterschiedlichsten Bereichen, die Sie selbst eingeben. **Fragen zum Spiel:** 300 Fragen aus den Bereichen Naturwissenschaften und Allgemeinwissen für das Gesellschaftsspiel. **Grafik-Tool:** Grafikspeicherprogramm legt auf Maustastendruck die aktuelle Bildschirmgrafik im IFF-Format auf Diskette ab. **Disk-Utility:** Kopieren, Umbenennen und Löschen von Dateien sowie das Formatieren und Duplizieren von Disketten über eine komfortable Benutzeroberfläche. **Sound-Digitizer:** Schnell und einfach digitalisieren Sie jeden beliebigen Klang mit der Steuersoftware. Viele Funktionen und Effekte stehen dafür zur Verfügung. Mit vielen Demonstrationsklängen zum Experimentieren. Mit dieser Zusatz-Software arbeitet der Sound-Digitizer (Bauanleitung in 68000er Ausgabe 1/87) auch mit dem Amiga 500 und dem Amiga 2000 einwandfrei zusammen. **Blitter-Demo:** Ein kleines Programm zur Demonstration des schnellen Amiga-Blitters. Fliegen Sie nach Belieben in einer dreidimensionalen Vektor-Landschaft umher.
1 Diskette für Amiga

Bestell-Nr.: 33707 **DM 29,90*** sFr 24,90*/öS 299,-*

Endlich: Pixelgenaue Hardcopies

Drucker-Utility: Druckroutine zur pixelgenauen Ausgabe beliebiger IFF-Grafiken auf Epson-kompatiblen Druckern. **Terminalprogramm:** Amiga-Basic-Programm, das alle Grundfunktionen professioneller DFÜ-Software beinhaltet. **Schnelles Kopierprogramm:** Mit diesem C-Programm sind weniger Diskettenwechsel beim Kopieren mit nur einem Laufwerk notwendig. **CLI-Tool:** Die Bedienung des Amiga-CLI erleichtern drei Amiga-DOS-Dateien in der RAM-Disk. **Sprachausgabe:** Zahlen bis zu einer Größe von einer Milliarde spricht dieses Programm perfekt in Deutsch aus. **Betriebssystem-Offset:** Alle Offsets der Amiga-Betriebssystem-Routinen für Besitzer des SEKA-Assemblers.
1 Diskette für Amiga

Bestell-Nr.: 32704 **DM 29,90*** sFr 24,90*/öS 299,-*

* Unverbindliche Preisempfehlung

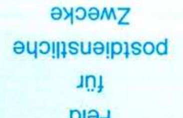
Sie suchen hilfreiche Utilities und professionelle Anwendungen für Ihren Computer? Sie wünschen sich gute Software zu vernünftigen Preisen? Hier finden Sie beides!
Unser stetig wachsendes Sortiment enthält interessante Listing-Software für alle gängigen Computertypen. Jede Woche erweitert sich unser aktuelles Angebot um eine weitere interessante Programmsammlung für jeweils einen Computertyp.
Wenn Sie Fragen zu den Programmen in unserem Angebot haben, rufen Sie uns an: Telefon (089) 4613-640 oder (089) 4613-133.

Bei Fragen zu Bestellung und Versand der Programmservice-Disketten wählen Sie bitte Telefon (089) 4613-232. Bestellungen bitte an: Markt&Technik Verlag AG, Unternehmensbereich Buchverlag, Hans-Pinsel-Straße 2, D-8013 Haar, Telefon (089) 4613-0. Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. Österreich: Ueberreuter Media Handels- und Verlagsgesellschaft mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0. Bestellungen aus anderen Ländern bitte nur schriftlich an: Markt&Technik Verlag AG, Abt. Buchvertrieb, Hans-Pinsel-Straße 2, D-8013 Haar, und gegen Bezahlung einer Rechnung im voraus.

Bitte verwenden Sie für Ihre Bestellung und Überweisung die abgedruckte Postgiro-Zahlkarte, oder senden Sie uns einen Verrechnungs-Scheck mit Ihrer Bestellung. Sie erleichtern uns die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

Einlieferungsschein/Lastschriftzettel
(nicht zu Mitteilungen an den Empfänger benutzen)

Gebühr für die Zahlkarte
(wird bei der Einlieferung bar erhoben)
bis 10 DM — — — 90 Pf
über 10 DM (unbeschränkt) 1,50 DM
Bei Verwendung als Postüberweisung gebührenfrei!



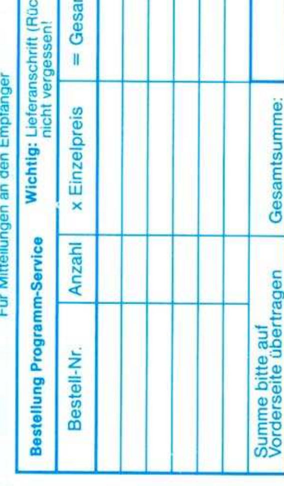
Bedienen Sie sich der Vorteile eines eigenen Postgirokontos
Auskunft hierüber erteilt jedes Postamt

Feld für postdienstliche Zwecke

Hinweis für Postgirokontoinhaber:
Dieses Formblatt können Sie auch als Postüberweisung benutzen, wenn Sie die stark umrandeten Felder zusätzlich ausfüllen. Die Wiederholung des Betrages in Buchstaben ist dann nicht erforderlich. Ihren Absender (mit Postleitzahl) brauchen Sie nur auf dem linken Abschnitt anzugeben.
1. Abkürzung für den Namen Ihres Postgiroamts (PGiroA) siehe unten
2. Im Feld »Postgiroteilnehmer« genügt Ihre Namensangabe
3. Die Unterschrift muß mit der beim Postgiroamt hinterlegten Unterschriftsprobe übereinstimmen
4. Bei Einsendung an das Postgiroamt bitte den Lastschriftzettel nach hinten umschlagen

Abkürzungen für die Ortsnamen der PGiroÄ:
Bln W = Berlin West, Kln = Köln, Dtmd = Dortmund, Lshfn = Ludwigshafen am Rhein, Esn = Essen, Mchn = München, Ffm = Frankfurt am Main, Nbg = Nürnberg, Hmb = Hamburg, Sbr = Saarbrücken, Han = Hannover, Stgt = Stuttgart, Krlh = Karlsruhe

Für Mitteilungen an den Empfänger



Bestellung Programm-Service			Wichtig: Lieferanschrift (Rückseite) nicht vergessen!	
Bestell-Nr.	Anzahl	x Einzelpreis	= Gesamtpreis	
Summe bitte auf Vorderseite übertragen		Gesamtsumme:		

Amiga-Tuning

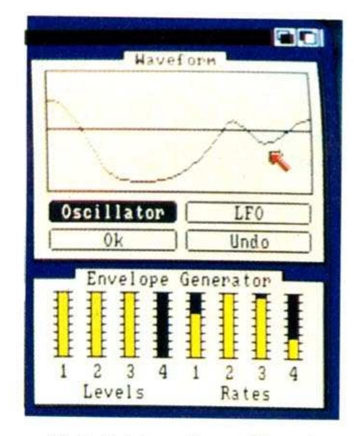
Wie läßt sich der Traumcomputer Amiga noch schneller machen? »Turbo« heißt das Zauberwort. Die Turbokarte beschleunigt den Amiga auf ungeahnte Geschwindigkeiten. Mit dem leistungsfähigen 68020-Prozessor, dem zusätzlichen, für hohe Rechengeschwindigkeit sorgenden 68881-Arithmetik-Prozessor für schnelle Floatingpoint-Berechnungen und wahlweise 14 oder 20 MHz Taktfrequenz wird aus dem schon nicht langsamen Amiga ein Computer, der sich mit einer wesentlich teureren Workstation messen kann. Die Kompatibilität zum Standard-Amiga bleibt weitgehend erhalten. Zusätzlich gibt es speziell angepaßte Software für den Turbo-Amiga. Ein Amiga zum Träumen.



Da ist Musik drin

Das wissen die Musiker unter den Amiga-Fans schon längst. Kein anderer Mikrocomputer kann Sounds mit einer solchen Tonqualität erzeugen. Gute Musikprogramme holen ein Maximum aus der Amiga-Soundhardware heraus. Machen Sie sich ein Bild vom aktuellen Angebot durch unsere Marktübersicht Musikprogramme.

Mit »Dynamic Drums« und »Audio Master« stellen wir Ihnen zwei Programme ausführlich vor. Dynamic Drums ist besonders interessant für die Spezialisten am Schlagzeug, die ihre Kenntnisse auch auf dem Amiga umsetzen möchten. Wenn Sie jedoch eher darauf aus sind, Natur-Sounds zu digitalisieren und dann daraus fantastische Effekte selbst zu kreieren, kommt der Audio Master gerade richtig.



Vielleicht wollen Sie aber erst einsteigen in die Welt der Computermusik. Zwei Grundlagenartikel über die Arbeitsweise von Musikprogrammen und den Übertragungs-Standard MIDI vermitteln Ihnen notwendiges Wissen.

Wer mit dem Amiga musizieren will, darf die nächste Ausgabe nicht verpassen.

Die 24-Nadel-Drucker kommen

Die 24-Nadel-Drucker erobern den Markt im Bereich um 1000 Mark. Drei Drucker machen den 9-Nadlern in diesem Preissegment Konkurrenz: der NEC Pinwriter 2200, der Epson LQ 500 und der Seikosha SL-80 AI. 24 Nadeln drucken schneller und in einer besseren Auflösung. Doch die drei Drucker bieten zusätzlichen Komfort. Jeder von ihnen kann mit besonderen Fähigkeiten aufwarten: Papierparkfunktion, mehrere eingebaute NLQ-Schriftarten, automatischer Papiereinzug, mehrere Emulationen sind nur einige der Besonderheiten. Welche Fähigkeiten jeder einzelne der Drucker besitzt, und wie das Triumvirat sich am Amiga schlägt, erfahren Sie in einem ausführlichen Vergleichstest.

VORSCHAU 2/88

AUSSERDEM IN DER NÄCHSTEN AUSGABE:

- AUSFÜHRLICHE TESTS VON SILVER, MAXIPLAN 500 UND MAXIPLAN PLUS, BUTCHER 2.0, FACC II UND TIMESAVER
- GO 64! — WIE GUT IST DER 64er-EMULATOR?
- WAS GIBT ES AN LERNPROGRAMMEN AUF DEM AMIGA
- JEDE MENGE LISTINGS ZUM ABTIPPEN IN C UND BASIC
- KURSE ÜBER GRAFIK, CLI UND DIE SPRACHE C

Die nächste Ausgabe erscheint am 27.1.1988 Erhältlich bei Ihrem Zeitschriftenhändler



Werkzeuge: Hilfe für jedermann

Wer den Amiga benutzt, benötigt früher oder später auch Werkzeuge, die ihm bei seiner Arbeit helfen. Damit sind Hilfsprogramme im Fachjargon gerne auch als Tools und Utilities bezeichnet, gemeint. Diese Programme, die gedacht sind, die Arbeiten mit dem Computer zu erleichtern, gehören eigentlich in jede Programmsammlung. Wir stellen Ihnen die besten und wichtigsten Tools und Utilities für den Amiga vor. Dabei finden nicht nur professionelle, sondern auch die zu sehr günstigen Preisen (unter 10 Mark) zu erhaltenden Freesoft- oder auch Public Domain-Programme Berücksichtigung. Außerdem vergleichen wir die besten Kopierprogramme. Die verschiedenen Fähigkeiten und natürlich auch die Geschwindigkeit werden herausgestellt.

Unsere

SPIELE DES JAHRES

für den

COMPUTER DES JAHRES





EMETIC SKIMMER

von Joachim Deeg

Wir schreiben das Jahr 5354. Die Erde hat soeben den ersten Computer mit künstlicher Intelligenz fertiggestellt und hat ihn auf dem fernen Planeten Pluto stationiert. Aber die Wissenschaftler haben die Fähigkeiten des Computers unterschätzt. Mit seinem immensen Wissen und seinem fehlenden Gewissen stellt er eine ernstzunehmende Gefahr für das gesamte Sonnensystem dar. Ihre Aufgabe ist es nun, den Emetic Skimmer, ein Raumschiff mit Spezialausstattung, durch die Verteidigungslinien des Computers zu steuern und ihn am Ende zu zerstören. Das neueste Superspiel für den Amiga - 2 Disketten randvoll gepackt mit Spitzen-Grafik, Digisounds und jeder Menge Action.







© SE-DESIGN 0241-151034

MIKE - THE MAGIC DRAGON

von Frank Neuhaus

Mike ist ein kleiner Drachenjunge, der von dem bösen Professor Dragan Drachenklau entführt worden ist, denn dieser benutzt in seinen Burgen und Schlössern mit Vorliebe kleine Drachen für seine garstigen Experimente. Doch Mike hat sich von den Ketten befreien können und sucht nun einen Weg durch das Labor des Professors, der in die Freiheit führt. Ein Kletter- und Springspiel der Extraklasse mit vielen neuartigen Extras.









PHALANX II - THE RETURN

nach Oliver Czesla

Nach dem Riesenerfolg des Actiongames PHALANX ist der langersehnte Nachfolger jetzt fertig. Und das Warten hat sich gelohnt, denn Grafik, Sound und Action haben eine weitere Steigerung erfahren. 1 oder 2 Spieler (gleichzeitig!) rasen über die vertikal scrollende Landschaft auf der Suche nach immer neuen Formationen der Gegner.



SIE GLAUBEN KEINEN WERBESPRÜCHEN?!? DANN LESEN SIE SELBST, WAS DIE DEUTSCHE FACHPRESSE ÜBER KINGSOFT-SPIELE SCHREIBT:

CHALLENGER von Andreas von Lepel

„Challenger ist kein Spiel, das man zur Spitzenklasse zählen kann, trotzdem ist es wegen seines relativ niedrigen Preises zu empfehlen." (ASM) **29.95 DM**

CITY DEFENCE von Uwe Kühner

„Originalgetreue Grafik, guter Sound, fantastische Steuerung und, was vor allem wichtig ist, eine gute Spielatmosphäre. Da es für den Amiga im Moment außerdem an guten Action-Spielen mangelt, kann man City Defence also nur wärmstens empfehlen, denn es ist wirklich Klasse!" (ASM) **29.95 DM**

CRUNCHER FACTORY von M. Aberle

„Vor allem zu zweit macht Cruncher Factory Spaß und angesichts des sehr attraktiven Preises kann man das Spiel empfehlen." (HAPPY COMPUTER)
„Eine amüsante und schnelle Pacman-Variante ist Cruncher, der zudem noch mit einem Level-Editor versehen ist. ... Die Grafik ist ausreichend gut, die digitalisierte Titelmusik wieder einmal hörenswert. Wer diesen Klassiker gerne für den Amiga haben möchte, sollte die Gelegenheit wahrnehmen." (KICKSTART) **29.95 DM**

DEMOLITION von Martin Aberle

„Von der Grafik her ist das Spiel recht gut, auch der digitalisierte Sound ist hörenswert." (KICKSTART) **29.95 DM**

EMERALD MINE

von Klaus Heinz & Volker Wertich

„Emerald Mine ist zur Zeit eines der besten (wenn nicht sogar das beste!) für den Amiga erhältliche Spiel. Da soll noch mal einer behaupten, daß in Deutschland keine guten Spiele produziert werden. Emerald Mine hebt sich von den bisher vorherrschenden Shoot 'em up Games deutlich ab und setzt in der Amiga-Softwareszene neue Maßstäbe. Dafür sorgt auch der unglaubliche Preis von nur knapp 30.- DM." (AMIGA SPECIAL)
„Die Steuerung des (oder der) Männchen(s) erfolgt mit dem Joystick und ist sehr exakt. So richtig Spaß macht das Game dann, wenn man zu zweit spielt. Zum Programm Emerald Mine kann man Kingsoft nur gratulieren, was einem bei diesem Spiel alles geboten bekommt, das sucht seinesgleichen. Nimmt man noch den überaus günstigen Preis von ca. 30 DM, dann ist mit Emerald Mine ein richtiger Hit entstanden, der jede Spielesammlung bereichert." (ASM)
„Die flotte Grafik mit witzigen Monster-Sprites und vor allem der Sound gefallen mit sehr gut: Die fetzige Titelmusik und die digitalisierten Sound-Effekte sind exzellent. Das Verblüffendste an diesem tollen Spiel ist aber der sehr niedrige Preis." (HAPPY COMPUTER)
„Grafik und Scrolling dieses Spieles sind überzeugend. ... Insgesamt hinterließ Emerald Mine einen guten Eindruck." (KICKSTART) **29.95 DM**

FLIP FLOP Amiga-Version von Edgar Törnig

„Das Programm ist für meinen Geschmack fast schon zu gut. Ich will ja nicht von mir behaupten, daß ich gut Reversi spielen kann, aber selten zuvor bin ich mit solch fliegenden Fahnen untergegangen." (ASM) **29.95 DM**

FORTRESS UNDERGROUND

von Andreas von Lepel

„Die Steuerung des Hubschraubers ist sehr exakt, in dieser Hinsicht ist das Programm gelungen. Auch das Scrolling geht in Ordnung. Der Sound kann sich hören lassen." (ASM) **29.95 DM**

KARATE KING von Marc Ebner

„Im Vergleich zu den beiden anderen Karatespielen ist Karate King das schnellste." (KICKSTART) **49.95 DM**

PINBALL WIZARD

von Hans Georg Berg

„Computer-Flippern stehe ich eigentlich skeptisch gegenüber, weil der Spielablauf größtenteils vom Zufall abhängt. Pinball Wizard ist aber so clever gemacht, daß er mich zu längeren Flipper-Sessions verführen kann. Die schön gezeichnete Grafik und die Digi-Sounds können überzeugen." (HAPPY COMPUTER)
„Die Idee, auf einem Homecomputer einen Flipper zu programmieren, ist zwar nicht neu, aber ich glaube, daß dem Programmierer von Pinball Wizard ein großer Wurf gelungen ist. ... Pinball Wizard ist die realistischste Umsetzung dieses Spiels, die ich kenne. ... Die Grafik, die dem Spieler geboten wird, ist vom Feinsten. ... Auch nach längerer Spieldauer konnte ich keine Schwachpunkte entdecken, so daß ich dieses Programm unbedingt empfehlen kann. Für mich ist Pinball Wizard ein Spiel, das in keiner (Amiga-) Sammlung fehlen darf." (ASM) **49.95 DM**

QUIWI Amiga-Version von Andreas Arens

„Fazit: ein sehr gutes Computer-Gesellschaftsspiel mit Zukunft." (HAPPY COMPUTER)
„Sogar die zehnte Revancherunde macht noch Spaß, denn 4000 Fragen machen Wiederholungen selten. QUIWI ist ein amüsantes Quizspiel für Feste und Familienfeiern." (RUN)
„QUIWI hingegen ist ein wirklich spaßiges wie unterhaltsames und lehrreiches Programm, wenn man mir diese Wiederholung von vorher verzeihen mag. Empfehlenswert!" (ASM)
„Ein abwechslungsreiches Spiel für die ganze Familie." (CHIP) **49.95 DM**

SOCCER KING

von Wolfgang & Helmut Theis

Fußballspiel für 1/2 Spieler; noch nicht getestet. **49.95 DM**

SPACE BATTLE von Andreas Hommel

„Space Battle ist ein solide gemachtes Spiel, das trotz der alten Spielidee viel Spaß bringt. Wenn man bedenkt, daß sowohl Cruncher Factory als auch Space Battle unter 30 Mark kosten, dann fällt einem die Entscheidung zum Kauf nicht mehr schwer. ... Beide haben, obwohl sie billig sind, mehr zu bieten als ähnliche Spiele." (AMIGA-MAGAZIN)
„Für knapp dreißig Mark erhält man mit Space Battle zwar kein neuartiges Spiel. Trotzdem ist es, besonders im Competition-Modus, spannend." (KICKSTART) **29.95 DM**

TYPHOON

von Christoph Sing & Rolf Wagner

„Auf diesem Hintergrund tummeln sich Massen von Sprites, die hervorragend ausgearbeitet und animiert sind. Es macht wirklich Laune, die verschiedenen Varianten von Raumschiffen und Hindernissen zu betrachten. ... Trotz einiger taktischer Varianten ist TYPHOON ein reinrassiges Actionspiel, das die Fans dieses Genres begeistern dürfte. Besonders die 50 verschiedenen Screens mit der erstklassigen Grafik und den blitzsauber animierten Sprites garantiert Freunden des Actionspiels lange Spielmotivation. ... Tolle Grafik und viel Action in TYPHOON ergänzen sich zu einem unterhaltsamen Spiel." (68000er)
„Wenn mich zukünftig jemand fragt, was ich unter einem guten Action-Spiel verstehe, werde ich ihn an TYPHOON verweisen. Die Handlung ist zwar dürftig bis nicht vorhanden, aber die Grafik ist ungeheuer faszinierend. ... Die Asteroiden-Felder, der Dschungel und die Stadt haben mich sofort begeistert. Allein die Neugierde, welche Sprites in der nächsten Angriffswelle kommen, hat mich vor den Monitor gebannt. ... Die Grafik steht so manchem Spielautomaten in nichts nach. ... Das Scrolling ist samtweich, Sprites sind reichlich vorhanden und gut animiert. Der Schwierigkeitsgrad des Spiels steigert sich ganz allmählich und die Idee mit den drei Extra-Waffen weist die Monotonie in die Schranken, die sonst nach einer Weile bei Ballerspielen entsteht. Die Präsentation kann sich auch sehen lassen: Schickes Titelbild, digitalisierte Disco-Rhythmen und eine prächtige High-Score-Liste." (HAPPY COMPUTER) **49.95 DM**

WILLY THE KID von Peter Meyer

Deutsches Text-/Grafik-Adventure; noch nicht getestet. **29.95 DM**



Bei uns gibt's jeden Monat Neuerscheinungen für Ihren AMIGA!

SPITZEN-SOFTWARE

KINGSOFT

MADE IN GERMANY

Alle Spiele selbstverständlich mit ausführlicher deutscher Anleitung, Super-Grafik und fantastischen Digisound-Effekten. Grauimporte haben keine deutsche Anleitung!

F. SCHÄFER · Schnackebusch 4
D-5106 ROETGEN ☎ 02408/5119
(nicht aufgeben!) Fax 02408/5213

KINGSOFT-Coupon

☐ Bitte senden Sie mir Ihren aktuellen Katalog

Meine Adresse ____________